DECEMBRE 1992
JANVIER 1993
NUMERO 82

Le numéro 40 F

A PROPOS DU CLUB

Le Bureau	Editorial	*1*
J. Belin	Ils ont fait PPC Paris	*2*
J. Belin	La réunion anniversaire, derniers échos	*3*
J. Belin	Domptez vos Virus	*4*
	Courrier du Coeur	*8*

HP-28

L. Grand	Une Horloge sur HP28	*10*
P. Heilbronn	Choc en retour	*14*

HP-48

G. Toublanc	Notes de lecture : *HP48 en Prépa*	*16*
L. Grand	Annuaire Téléphonique	*22*
G. Toublanc	Fast-orielles (Acte II)	*23*
G. Toublanc	Test de rapidité d'un programme	*26*
G. Toublanc	Critiques et réponses	*27*
G. Toublanc	Errare Toublancum est	*28*

HP-95

J. Belin	Du Saturn au 8086 (Acte III)	*32*

LE COIN DES CODES *37*

EDITORIAL

Bien que notre réunion anniversaire soit programmée fin Janvier (pour permettre à tous les participants de la réunion anniversaire de HPCC à Londres de récupérer), ce mois de décembre 1992 marque réellement le dixième aniversaire de PPC Paris. En effet, notre association a vu son existence débuter officiellement le 2 décembre 1982. Dans ces temps là, la HP-41 règnait en maître dans le petit monde des utilisateurs de calculateurs HP. Ensuite, Hewlett-Packard commercialisa de nouvelles machines, dont certaines, portant les doux numéros de 71 ou 28 contribuèrent à perpétuer une tradition tournée vers l'innovation, au grand plaisir de nombreux utilisateurs ne jurant plus que pour la firme aux deux lettres. Pendant ce temps, la HP41 atteignait tranquillement ses dix ans de production ininterrompue, ce qui restera peut-être longtemps le record absolu. Depuis, deux nouvelles machines, les HP48 et HP95, se disputent le marché, et les nouveaux utilisateurs rejoignent les rangs des passionnés de longue date...

Pendant ce temps, PPC Paris connaissait un succès grandissant, recrutant ses membres aussi bien dans les milieux étudiants que chez les professionnels de tous les domaines, habitant aussi bien le centre de Paris que la banlieue d'une ville située au centre de l'Argentine. Grace à cette association de passionnés, de nombreux adhérents ont appris à mieux se servir de leur machine, puis ont redistribué leurs connaissances aux autres membres. Ceci a permis au club, par exemple, d'occuper le premier rang mondial dans le domaine de l'assembleur Saturn, sur HP-71.

Cependant, comme beaucoup d'autres clubs, PPC Paris a rencontré une période noire, mais, là où des clubs apparement plus solides ont échoué, notre club a su traverser cette épreuve et en est sorti peut être quelque peu affaibli, mais toujours vivant.

Aujourd'hui, après s'être regroupé autour d'un groupe de fidèles (la moitié des adhérents actuels appartiennent au club depuis plus de 5 ans), PPC Paris a repris un rythme croissant de nouveaux arrivants et peut donc voir l'avenir sous un meilleur jour. Cependant, pour retrouver un club aussi grand et puissant que par le passé, il est nécessaire que les membres reprennent l'habitude de communiquer leurs expériences et leurs programmes. Pour cela, il n'existe qu'un seul moyen universel de communication entre les adhérents : JPC. Que vous soyez débutant ou spécialiste, que vous soyez prêts à faire des articles de 10 pages, ou n'avez le temps que de faire des petites contributions de 40 lignes, n'ayez pas peur. Aucun membre n'a jamais ri d'un article d'un autre adhérent, quelque soit les différences de niveaux entre ces deux personnes, car il a toujours su qu'il y aurait toujours d'autres adhérents pour qui cet article serait parfaitement adapté. *JPC* est la vitrine du club. Notre travail est de la faire briller, mais c'est surtout à vous de la remplir, afin que des passants regardant au travers aient envie d'y entrer et y restent jusqu'à notre vingtième anniversaire...

En attendant de nous retrouver, dans le prochain *JPC* ou à notre réunion anniversaire, nous vous souhaitons un joyeux Noël et une nouvelle année pleine de beaux petits programmes !

Le bureau.

ILS ONT FAIT PPC PARIS

Directement ou indirectement, de nombreuses personnes et sociétés ont collaboré avec notre club ou ont contribué a son histoire. Je vais donc tenter, au nom du bureau actuel, de tous les remercier. Si par le plus grand des hazards j'avais oublié de citer quelqu'un, qu'il me le pardonne. Attachez vos ceintures, c'est parti !

- En premier lieu, Richard Nelson, qui a créé le premier club PPC (aux USA) et Phillipe Guez, qui a créé PPC Paris.

- Thomas Affinito, Yves Alajouanine, Robert Amram, Lionel Ancelet, Pierre Antoine, Asdin Aoufi, Olivier Arbey, Philippe Aspéro, Robert Atlan, Jean-Louis Attenoux, Luzius Auer, Christian Bachelet, Christian Bacquet, Jean-Marc Baillard, Wolfgang Baltes, Stéphane Barizien, Frédéric Barnaud, Jean-Paul Barre, Pierre Bassaler, Jacques Baudier, Jean-Claude Becker, Mario Benedetti, Thierry Besancon, Mathieu Besson, Xavier Bille, Alain Bochet, Alexandre Boldireff, Jean-Pierre Bondu, Jurgen N. E. Bos, Jean Boschat, Thierry Bravier, Alexandre Buchmann, Jan Buitenhuis, Philippe Canuel, Denis Castelain, Roger Charpentier, Ronic Chiche, Laurent Chouraki, Michel Clabot, Pierre-Jérome Clémenceau, Pierre Colignon, Daniel Connan, Paul Courbis, Mark Cracknell, David Dalila, Olivier Dancer, Philippe Davase, Pierre David, Damien Debril, Eric Delagnes, Jérome Devémy, Jean-Jacques Dhenin, René Dine, Jean-Daniel Dodin, Yann Dolhen, Claude Dupré, Jacques Durand, François Duret-Lamouroux, Peter Ehrenberg, Jack Elhay, Henrick Elnaes, Alain Farge, Gigi Filippini, Craig Finseth, Jean-Claude Foures, Francis Friesse, Olle Galmo, Jean-Francois Garnier, Eric Gengoux, Gabriel Gil, Bruno Gil, Alain Gillet, Frédéric Gobin, Christophe Gottheimer, Alain Goubault de Brugière, Laurent Grand, Philippe Guez, Lionel Guillou, Tony Guilloux, Franck Guy, Phillipe Heilbronn, Vincent Herlicq, Alain Herreman, Jean-Yves Hervé, Hewlett-Packard, J. Hulaas, Laurent Istria, Daniel Jacob, Ulrich Jansen, Christian Jegouzot, Didier Jehl, Laurent Jolia-Ferrier, Jean-Michel Kefaloucos, Franklin Khazine, Dang-Trung Khoi-Nguyen, Volker Klann, Christoph Klug, Gérard Kossmann, Henri Kudelski, Jean-Claude Kursner, Sébastien Lalande, Pierre Langlois, Roger Le Bris, Francois Le Grand, Guillaume Le Stum, Franck Lebastard, Christopher Lishka, Arne Lührs, Jean Maille, Gérard Mangeney, Claude Marcoin, Michael Markov, Nicolas Martin, Jean Martinelli, Michel Martinet, Michel Martinet, Michel Maupoux, Wlodeck Mier-Jedrzejowicz, Lucien Monet, Morando, Jean-Jacques Moreau, Bernard Morisseau, Christian Morlot, Jean-Paul Nalin, Jean-Yves Naour, Richard Nelson, Philippe Nicodème, Daniel Odos, André Oisel, René Ouvray, Jean-François Pelanne, R. Pennetier, Steen Petersen, Stefano Piccardi, Pierre Picheret, Aimé Pierrard, Olivier Pilloud, Michel Polski, B. Pons, Olivier Pougeon, Etienne Poupée, Frédéric Poupon, Robert Pulluard, Jean Reibel, Philippe Romascano, Francis Rosange, A. Rosset, Aurel Rottman, Yann Rousse, Daniel Saada, Jean-Pierre Sandoz, Robert Schwartz, Jake Schwartz, Laurent Serano, Megha Shyam, Jean-Marie Simon, Franck Stengel, Jannick Taillandier, Dominique Talon, Tapani Tarvainen, Philippe Tenand, Stefano Tendon, Hervé Thévenon, Lewis Thomas, Gilbert Tisserand, Guy Toublanc, Jean-Pierre Toyre, Bruno Tredez, Frédéric Vadez, Christophe Vaillant, Ludovic Valois, Fred Van Der Windt, Jacques Vaucelle, Serge Vaudenay, Frank Wales, Brian Walsh, Michel Weil, Franck Wettstein, William Wickes, Bertil Wicklund et Raan Young qui ont fait publier au moins un article dans les 3578 pages de notre journal !

- Lionel Ancelet, Olivier Arbey, Jacques Baudier, Xavier Bille, Jean-Claude Becker, Laurent Chouraki, Pierre David, Vincent Delorme, Jean-Jacques Dhenin, Pierre Franck, Eric Gengoux, Philippe Guez, Laurent Istria, Daniel Jacob, Michel Martinet, Jean Reibel, Aurel Rottmann et Jannick Taillandier qui, par leur participation aux différents bureaux, ont souvent dû faire un nombre incalculables d'heures supplémentaires, afin de faire fonctionner le club et mettre le journal en pages !

- Les clubs Français et étrangers : PPC Toulouse, PPC Lausanne, PPC Danemark, PPC Australie, PPC Norvège, HPCC, Prompt HP-GC, PCX Brugge, HPX, CCD et STaK, qui nous ont très souvent servi d'exemple.

- Les familles des personnes précédement citées, qui ont souvent dû supporter une passion dévorante !

- Les sociétés Fogebur, Mistral Photo, Copy Express et Paris Copie qui ont souvent dû imprimer nos *JPC* dans des delais records !

- La (regrettée) Règle à Calcul, Maubert Electronic, Compta France et EduCALC qui ont souvent permis à nos adhérents, grace aux réductions, de s'approvisionner en matériel à moindre frais !

- Les sociétés Zengrange, W&W Software, Eramco, WM Products, Krystal, CMT, TDS, Sparcom, TDS, ACE... qui ont fait d'excellents modules ou périphériques, alors que Hewlett-Packard avait la tête ailleurs !

- William Wickes, Keith Jarett, Jim Donnelly, Richard Arvey, Jean-Michel Ferrard, Paul Courbis et Sébastien Lalande... qui nous ont écrit des livres faisant maintenant figure de référence !

- Les développeurs de Corvallis, qui nous ont créé nos HP-41, HP-71, HP-28, HP48, HP95 et autres, avec une pensée particulière au programmeur anonyme qui a laissé passer une cartaine bogue dans la Rom de la HP-41, nous laissant ainsi découvrir la programmation synthétique qui a été la cause principale de l'éclosion des clubs HP dans le monde !

- Nos contacts chez HP France : Philippe Chaillot, Robert Bayle, Eric Clément et Michel Maupoux avec qui nous avions d'excellents contacts et maintenant Jean-Paul Barnier, Corrine Brizard et Michel Serge avec qui nous espérons que la collaboration sera encore meilleure.

Et enfin, ceux sans qui rien de tout cela n'aurait existé : William Hewlett et David Packard !

Jacques Belin (123)

BIENTOT, LA REUNION

Ce petit article a pour but de vous donner quelques informations complémentaires sur notre réunion exceptionnelle des 30 et 31 Janvier.

Tout d'abord, il est fort probable qu'au cours de notre assemblée générale, nous fassions un appel pour remplacer un ou plusieurs membres du bureau. Si le travail en lui même n'est ni important ni compliqué, il est nécessaire que les candidats habitent la région parisienne afin que nous puissions assurer un contact direct (ne serait-ce que pour nous transmettre rapidement certains documents) quand cela est nécessaire. Si vous êtes volontaires, contactez moi avant l'assemblée.

En ce qui concerne les prix du tournoi de programmes, sachez que le premier prix sera un HP95LX (version 1 Méga Octets), offert par HP France (merci Jean-Paul Barnier !). Les prix suivants seront déterminés après le tournoi, car il s'agira généralement de périphériques dédiés à une machine précise (cartes mémoires...). En effet, il serait regrettable de donner une carte HP48 à un utilisateur de HP95 ! Nous sommes en négociation avec quelques sociétés pouvant nous offrir ces prix, mais sachez déjà que Maubert Electronic et Palmsoft nous ont déjà promis de participer !

Un IBM PC devrait être disponible en libre service. Vous trouverez dans son disque dur de nombreux programmes du domaine public pour les HP48 et HP95, ainsi que ceux qui sont parus dans les JPC. Bien sûr, vous serez tout à fait autorisés à ajouter de nouveaux programmes sur le disque dur, afin d'en faire profiter les autres !

Wanted !

Nous comptons organiser, entre autres, exposition de tous les calculateurs fabriqués par HP ainsi que leurs périphériques (fabriqués par HP ou non) par HP. Nous avons déjà réuni de nombreux, mais certains modèles ou variantes nous manquent encore. En voici la liste (attention, les lettres sont significatives) :

01 (montre calculatrice), 01 GOLD, 10B, 10C, 11C, 14B, 17B, 17BII, 18C, 19B, 19BII, 20S, 21S, 22S, 25C, 27S, 28C, 28S, 28COM, 29C, 32S, 32SII, 33C, 37E, 37C, 38E, 41 OPT 001 (HP41 sans marquage de touches), 46, 48S, 81, 83, 91, 92, 94, 95C, 97S.

Pour les périphériques HP :

Interface vidéo 82163, Interface GPIO 82165, Interface HPIB 82169, MODEM acoustique 82168, Interface RS-232C 82164, module 82211A.... Ainsi que tous modules ou périphériques HP-41, HP-71, HP-48 et HP-95 fabriqués par HP et les autres sociétés (CMT, SPARCOM...).

Si vous avez un de ces éléments, contactez nous (même si vous ne pouvez pas venir à la réunion). Rassurez vous, nous vous rendrons ces éléments dès la fin de la réunion. Rappelez vous que notre but est de tenter de montrer l'exposition la plus complète possible.

Il est à noter à cette occasion que la société Compta France, par l'intermédiaire de Lionel Bardano, nous a offert récement quelques calculateurs : HP 33E, 22, 55, 65 et 35 ! Si deux exemplaires (33 et 55) sont des modèles d'exposition (sans électronique interne), les HP-22 et HP-35 ne demandent que des accumulateurs bien chargés pour fonctionner ! Que Compta France en soit remercié et espérons que cela ne soit que le début d'une grande collection !

N'attendez pas pour vous inscrire, car nous devons absolument avoir une estimation assez précise du nombre de personnes présentes, afin de ne pas avoir

de mauvaises surprises (dues à un trop grand nombre de participants non inscrits) au dernier moment !

Le programme de ces deux jours, ainsi que les noms des presonalités présentes devrait être envoyé prochainement aux adhérents (probablement avec la convocation à l'assemblée générale) et à ceux qui se seront inscrits.

Jacques Belin (123)

DOMPTEZ VOS VIRUS

Certaines personnes désirent participer à notre tournoi de programmes, mais hésitent encore, car elles pensent que ce genre de programmes est trop au dessus de leur niveau de programmation. Cet article a pour but de leur donner quelques conseils pour débuter leur analyse du programme. Ne le prenez cependant pas comme étant une référence absolue car, d'une part, si vous désirez gagner vous devrez échafauder votre propre tactique, et d'autre part, je n'ai tout simplement pas eu le temps de réellement travailler sur une analyse vraiment complète !

De toute façon, le seul but de cet article est de vous encourager à participer et non de vous en dissuader !

Conseils généraux

Tout d'abord, nous savons que ce jeu est relativement difficile à programmer. En fait, je le situe à mi-chemin entre Othello et le Go. Cependant, n'ayez pas peur que votre programme soit d'un trop faible niveau, tous les autres le seront aussi. Ceci est compréhensible, car le jeu étant totalement inédit et connu relativement peu de temps avant le concours, je suis prêt à parier ma place à la présidence du Club qu'aucun programme ne sera encore capable de vaincre un adversaire humain ! Il faudra peut-être de nombreux mois avant de voir des programmes réellement compétitifs. Cela n'a aucune importance, car le but du concours n'est que d'opposer des programmes entre eux sur un même sujet, même si ils sont d'un niveau très inférieur à un joueur humain. En fait, un programme qui vous semblera très moyen et dont les coups vous semblerons trop prévisibles pourra très bien mettre les autres en déroute, en utilisant un "petit quelque chose" que les autres n'auront pas, et arriver en premier !

Tout programme ne s'écrivant pas sans une bonne analyse préalable, je vous conseille impérativement de respecter les étapes suivantes, dans l'ordre et sans brûler les étapes :

1 - Jouez au maximum contre plusieurs adversaires humains (si par hasard vous vous ennuyez pendant le réveillon...). Ceci vous permettra de trouver les tactiques de jeu pouvant être intéressantes, ainsi que les pièges à éviter !

2 - Analysez le programme. Choisissez dans les tactiques que vous avez trouvé celles qui peuvent être le plus facilement formulées sous forme d'algorithmes, puis déterminez la façon dont seront codées les données et les fonctions de bases du programme en accord avec ces algorithmes.

3 - Une fois que vous connaitrez précisement la structure du programme, vous pourrez alors commencer à programmer.

4 - Testez votre programme aussi bien en jouant directement contre lui qu'en le faisant jouer contre d'autres adversaires humains ou des programmes. Ceci vous permettra de voir si il existe des failles grossières dans vos algorithmes, failles que vous pouvez très bien laisser passer si vous restez seul.

Ne perdez pas de temps à faire un programme avec une superbe présentation graphique. Cela n'est pas le but du jeu. Dans l'absolu, un simple échange de coordonées sans affichage du plateau de jeu suffirait, mais il est cependant conseillé de l'afficher, au moins en mode texte, afin de vérifier que tout se passe bien :

```
8 . . . . . . . .   Temps de reflexion :
7 . . X o . X x .    00:03:59
6 . . o o O . . .
5 . . . . o x . .   Mon coup :
4 . . . x x o . .    C7 (Virus)
3 . . . . x . . .
2 . . . . . . . .   Votre coup :
1 . . . . . . . .    __ _
  A B C D E F G H
```

Tactiques en vrac

Le but de cette partie est de vous donner quelques indications de tactiques qui peuvent être intéressantes d'utiliser. Evidement, dans votre intérêt, il vous faudra ajouter vos propres idées, car si tous les programmes n'utilisent que ce qui est marqué ci-dessous, le tournoi risque de manquer d'interêt ! D'autre part, ces conseils ne doivent pas être pris comme une règle absolue, car, comme tout nouveau jeu, les meilleures tactiques sont encore à trouver et

risquent fort de contredire mes premières conclusions.

- Choisissez une tactique de jeu offensive, car un joueur qui ne tenterait que de regrouper ses pions, sans tuer des bactéries adverses et en empêchant l'autre de tuer ses bactéries ne ferait, au mieux, qu'une partie nulle. N'oubliez pas qu'en cas d'ex-aequo, ce sera le programme qui aura gagné avec les plus grandes différences de pions qui remportera le tournoi.

- Tentez d'occuper la partie centrale du plateau de jeu, puis étendez vous dans toutes les directions.

- En contrepartie, si vous êtes dans une position de déséquilibre, avec seulement quelques pions placés sur un côté du plateau, il est indispensable d'ouvrir un deuxième front de l'autre côté du plateau, en y placant un Virus.

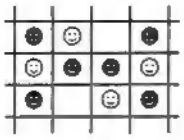
- Ne placez pas vos Virus trop tôt, car vous perdrez la possibilité de contrôler le jeu plus tard.

- Ne les placez pas non plus trop tard, car, à ce moment vous n'aurez peut-être plus une zone libre suffisante pour développer une colonie de bactéries.

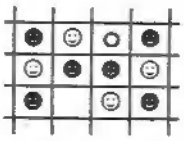
- En fait, il vaut mieux les placer une fois que l'adversaire a déjà placé ses Virus, car à ce moment là nous savons que l'adversaire ne peut plus ouvrir de nouveau front. Cependant, pour le concours, faites attention aux programmes qui penseront que vous utiliserez cette règle, et ne joueront pas leurs Virus dans le seul but de vous bloquer !

- Placez vos Virus sur des cases défavorables à des bactéries normales (par exemple, des cases déjà entourées d'au moins trois bactéries adverses).

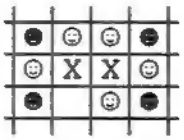
- Détectez les bactéries adverses ne cotoyant qu'une seule bactérie de sa couleur, puis regardez si cette dernière est en danger (entourée de trois bactéries de votre couleur). C'est celle là qu'il faut tuer, puisqu'elle rendra la première citée isolée, donc non viable ! En poussant un peu plus loin le raisonnement et en tentant de savoir si plusieurs bactéries sont dépendandes d'une ou deux bactéries, on peut s'apercevoir que bien placée, une seule bactérie peut faire des ravages, en tuant cinq ou six (peut être plus !) bactéries adverses ! Par exemple :



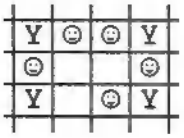
1) Position Initiale



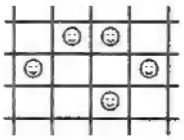
2) Blanc place la bactérie 'O'



1) Les bactéries 'X', entourées de 4 bactéries, sont tuées

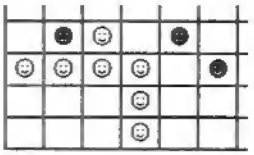


2) Les bactéries 'Y', isolées, sont aussi tuées



1) Configuration finale !

Une fois que les Virus adverses auront été placés, plutôt que de placer ses bactéries de façon compacte, il est conseillé de tenter de créer une zone libre autour d'un coin du plateau, limitée par deux lignes droites à 90 degrés. Par exemple, la configuration suivante :



permettra au joueur blanc de se réserver au moins 12 pions en fin de partie, en remplissant la zone vide qui ne peut plus être occupée par une bactérie adverse, puisque tous les Virus ont été joués. Il faut cependant protéger la bactérie placée à l'intesection des deux lignes, car si il est impossible de tuer une bactérie faisant partie d'une ligne droite (car, à moins de placer une bactérie de l'autre côté de cette ligne, il ne peut pas y avoir plus de trois bactéries adverses autour d'une bactérie), il est toujours possible de tuer celle qui fait l'intersection, ce qui peut permettre à d'autres bactéries de s'incruster dans la brêche.

Formulation du programme

Cette section contient des indications destinées à ceux qui n'ont jamais programmé ce genre de problème. Cependant, ne vous attendez pas à un cours d'intelligence artificielle, car je n'ai ni le temps, ni l'expérience nécessaire pour faire ce genre d'article. De plus, étant donné que la diversité des machines et languages pouvant être utilisés dans le tournoi, ces indications ne peuvent pas être trop précises. Ce sera donc à vous de définir en détail l'organisation de votre programme.

Formulation des données

Le plateau de jeu peut être modélisé par un simple tableau 8×8, contenant les différents pions posés par les joueurs, suivant une représentation telle que celle ci :

- Case vide : 0
- Case occupée par une bactérie de votre couleur : 1
- Case occupée par un virus de votre couleur : 2
- Case occupée par une bactérie adverse : -1
- Case occupée par un virus adverse : -2

Le fait de donner des valeurs négatives au pions adverses permet de simplifier les tests d'occupations de cases en exécutant des tests par rapport à zéro (égal, supérieur ou inférieur), qui sont souvent plus rapides que des tests sur une constante définie.

Créez une fonction de base ayant pour entrée les coordonnées d'une case précise et qui retourne globalement le type de bactérie placée sur cette case et le nombre de bactéries de chaque couleur entourant cette case. Si vous travaillez en langage utilisateur, cette valeur peut être, par exemple, de la forme `P.ja` où `P` est le type de pion occupant la case (0 si la case est vide), `j` le nombre de pions de votre couleur entourant cette case et `a` le nombre de pions adverse. Cette fonction vous permettra donc de dégrossir le travail en vous donnant les cases où vous ne pouvez pas jouer (soit parce qu'il y a trop de pions adverses autour, soit parce qu'il n'y a pas de pion de votre couleur à coté), ainsi que les cases occupées par un pion de l'un ou l'autre joueur et qui sont en danger (entourées de 3 pions), par exemple.

Faites attention, en écrivant cette fonction, de ne pas chercher à lire une case qui serait en dehors du plateau de jeu !

Si vous en avez la possibilité, écrivez cette fonction en assembleur, car c'est celle qui sera la plus fréquement utilisée, donc celle qui prendra le plus de temps au cours de l'analyse.

Début de la partie

Le règlement du tournoi interdisant les algorithmes possédant des routines aléatoires, le programme devra déterminer lui-même la position de départ.

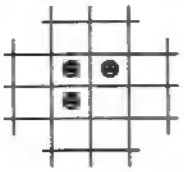
Il n'y a pas de problème de programmation pour le joueur noir, qui commence toujours, car vous pourrez coder directement dans le programme le coup (composé de la pose de 3 bactéries) que vous semblera le plus approprié, car vous l'aurez déterminé au cours des parties que vous aurez joué.

En ce qui concerne le coup suivant (le joueur Blanc pose 3 bactéries), je pense qu'il est beaucoup plus simple de passer par une petite bibliothèque d'ouverture. En effet, une petite analyse m'a montré qu'il n'y a que 84 façons (à moins que je ne me sois trompé) , pour le joueur noir, de placer ses bactéries en respectant les règles de vie du jeu Virus. Ceci veut dire que vous pouvez réduire le nombre de possiblités du joueur Blanc à 84 coups en créant un tableau comportant une combinaison de pions blancs pour chaque coup possible du joueur noir. Par exemple :

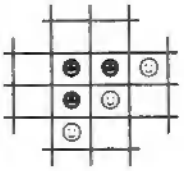
Coup Noir	Réponse de Blanc
D6-E6-C5	F4-D4-E5
D6-E6-D5	C5-C4-D4
D6-E6-E5	F5-E4-F4
.	.
.	.
.	.

Bien sûr, vous devrez vous même créer cette bibliothèque, en choisissant la réponse qui vous semblera la plus appropriée pour chaque combinaison du joueur Noir. Rassurez vous, compte tenu des symétries, il n'y a (apparement) que 12 cas différents.

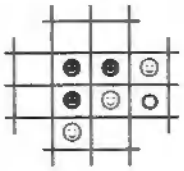
Faites cependant très attention à certaines combinaisons, qui peuvent vous être fatales. Par exemple :



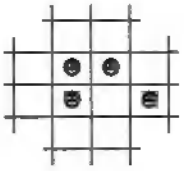
1) Noir Commence



2) Blanc Continue...



3) Noir Joue en 'o'



4) Résultat...
(Blanc sera contraint de jouer un Virus, puisque il n'a plus de bactérie vivante !)

L'interêt principal de cette bibliothèque d'ouverture est que le temps de réflexion pour le premier coup du joueur blanc est quasiment nul, puisque il peut se résumer à une simple recherche dans un tableau de 84 entrées, alors que si il avait fallu recourir à un algorithme spécifique, le problème aurait été beaucoup plus compliqué (recherche de toutes les combinaisons de 3 pions blancs compatibles avec les

Règles de vie, puis choix de la meilleure combinaison parmi celles-ci !).

Analyse des coups

La première étape de l'analyse consistera à compléter chaque case du tableau avec le résultat de la fonction que nous avons créé. Ceci permettra de simplifier la suite de l'analyse. Afin de diminuer le temps de réflexion, il peut même être possible de ne remettre à jour que les zones voisines des pions venant d'être joués.

Ensuite, vous tenterez de trouver, par exemple, pour chaque pion adverse entouré de 3 pions de votre couleur si il y a une case vide autour de lui. Si cette case est jouable pour vous, c'est une bonne occasion pour tuer une bactérie ennemie ! Vous pouvez bien sûr procéder de la même façon pour protéger vos propres pions.

Quel que soit le type d'analyse que vous ferez, le principe général sera de donner une certaine note à chaque coup possible. Une fois que vous aurez examiné toutes les possibilités, vous pourrez jouer la coup ayant la meilleure note.

Pour attribuer cette note, vous pourrez par exemple, décider d'un barème, par exemple en donnant 500 points à un coup tuant une bactérie adverse, 5000 points à un coup tuant trois ou quatre bactéries, 100 points à un coup normal, -1000 points à un coup plaçant une bactérie sur une case entourée de trois bactéries adverses.... Etant donné que chaque case peut présenter plusieurs de ces caractéristiques, une simple addition (ou une multiplication de coefficients correspondants) vous donnera la note recherchée. En ce qui concerne les case vides ou non jouables, une note minimum (-100000 par exemple) sera appliquée.

Contrairement à certains jeux (Othello, par exemple) il ne semble pas qu'il y ait de notion de cases "faibles" ou "fortes" (les coins, par exemples). Cependant, si vous désirez favoriser le placement des pions vers le centre du plateau, vous pouvez, par exemple, attribuer un coefficient multiplicateur de 2 aux cases centrales, 1.5 aux case suivantes et 1 aux cases formant la bordure du plateau.

Mise en jeu des Virus

Les Virus pouvant être joués n'importe quand et sur n'importe quelle case libre, mais étant présents en nombre limité, il est nécessaire de leur appliquer un traitement particulier.

Vous pouvez, par exemple, décider de l'opportunité de jouer un Virus si trop de pions de votre couleur sont placés dans une zone correspondant à un quart du plateau. Ceci peut être fait en comptant le nombre de vos pions dans chacun des quatre secteurs du plateau, puis en comparant ces nombres et jouer un virus si vous trouvez une différence dépassant un certain seuil.

Vous pouvez aussi décider, plus simplement, de jouer deux virus entre (par exemple) le 20eme et le 30eme coup, et conserver le dernier pour casser une zone trop protégée par le joueur adverse.

Prise des pions

La méthode la plus simple (et la plus sûre) pour effectuer la prise des pions est d'écrire une routine procédant en deux passes, après que le nouveau pion ait été posé :
1 - Balayage de toutes les cases et élimination de toutes les bactéries (hormis les Virus) entourées de 4 pions d'une même couleur.
2 - Nouveau balayage de toutes les cases et élimination des bactéries (toujours hormis les Virus) ayant été rendues isolées par l'élimination des bactéries précédentes.
A priori, cette routine est indépendante de la couleur des pions du joueur qui vient de jouer, et peut tout simplement utiliser la fonction que j'ai décrit précédement.

Annoncez le coup joué avant d'effectuer les différentes modifications du tableau (placement des pions, élimination des bactéries tuées...). Ceci vous permettra d'effectuer ces opérations sans que ce temps soit comptabilisé.

Pour terminer, je ne ferais que conseiller aux participants de la région parisienne de se rendre à notre réunion du 9 janvier, où nous pourrons peut-être comparer les premières versions de programmes et vous donner quelques derniers conseils.

Jacques Belin (123)

Addendum sur la Règle du Jeu de Virus, non notée dans l'article paru dans le précédent JPC : Les pions symbolisant les bactéries tuées sont replacées avec les pions non utilisés, et peuvent donc être remis en jeu.

COURRIER DU COEUR

Frédéric BOLDIREFF
25, rue Damesme
75013 PARIS
Tel : (1) 45 65 33 63

Vend :

HP-71, Lecteur de cartes + 100 cartes magnétiques : 1500 FF.

HP-28

L. Grand	Une Horloge sur HP28	*10*
P. Heilbronn	Choc en retour	*14*

HORLOGE POUR HP-28

Avant de vous présenter mon ensemble de programmes, je tiens à déplorer la disparition des articles pour HP-28 des derniers numéros de JPC. Quoique je ne possède plus de HP-28 (eh oui, les nombreux *Memory Lost* dûs au développement de programmes en assembleur m'ont incité à migrer vers une machine plus confortable), je pense que c'est encore une très bonne machine et qu'elle n'a pas que des désavantages par rapport à sa petite soeur (la HP-48), ne serait-ce qu'au point de vue du format (on la met facilement dans une poche, même assez petite) et du clavier alphanumérique séparé. Mais, rassurez-vous, j'en ai terminé de me lamenter et je passe immédiatement à ce qui nous intéresse.

Lorsque j'ai acheté ma première HP-28, je fus très vite enchanté par sa convivialité et par ses possibilités de manipuler des objets divers ainsi que des variables. Pourtant, une immense frustration ne tarda pas à se manifester : il manquait une horloge incorporée à ce superbe outil. J'ai dû patienter jusqu'au numéro 46 de *JPC*, dans lequel l'adresse d'une routine activement recherchée fut enfin dévoilée : cette routine renvoie la valeur d'un nombre hexadécimal incrémenté environ 8192 fois par seconde. Cette adresse n'était valable que pour les HP-28C version 1BB. Dans le *JPC* 53, Paul Courbis révéla l'adresse de la même routine pour les HP-28S version 2BB.

Dès lors, il fut assez facile d'écrire une simple horloge pour ces machines. Le défaut est que l'incrémentation ne se fait pas exactement tous les 8192ème de seconde. Plus encore, cette fréquence varie de machine en machine. C'est alors que je me suis inspiré du fabuleux module *TIME* de la HP-41 (eh oui, j'ai aussi eu une HP-41CX) pour créer une horloge "auto-correctible". Voici très brièvement le principe de fonctionnement du programme :

Le programme lit la valeur de ce nombre automatiquement incrémenté par la machine, le divise par un certain nombre (que j'ai appelé TFACT. Ceci assure une vitesse plus ou moins correcte de votre horloge. Il ne reste plus qu'à lui ajouter un autre nombre, delta pour obtenir une date et une heure justes.

J'ai choisi le nombre TFACT de telle façon que le résultat de la division citée plus haut soit incrémenté de un toutes les vingt-quatre heures. Une bonne valeur de départ pout TFACT est donc 24*3600*8192, soit 707'788"800. Le nombre delta quant à lui varie énormément de machine en machine, je ne peux donc pas en estimer arbitrairement la valeur.

L'ensemble de programmes que je vous propose consiste en fait en deux sous-ensembles de programmes : le premier est celui que je viens de décrire et constituera le répertoire TIME du menu HOME (si je peux m'exprimer ainsi, car HOME n'est pas un menu, mais un répertoire). Le second effectue des calculs sur des dates , sera stocké dans le sous-répertoire DATES du répertoire HOME (cette fois, c'est correct). Je vous prie de respecter ces deux noms de répertoire, car ils seront utilisés par les programmes pour passer de l'un à l'autre. Je commence par le répertoire le plus simple, c'est à dire DATES.

Le répertoire DATES

Toutes les dates doivent être tapées sous la forme d'un nombre réel mm.jjaaaa, où mm est le mois, jj le jour et aaaa l'année. Par exemple, le nombre réel 10.231987 représente la date du 23 octobre 1987 et vice-versa. Si cela vous paraît bizarre, vous vous apercevrez très vite que ce n'est qu'une question d'habitude.

Le répertoire DATES contient les objets suivants : MD BSX DAY DOW DDAYS et DATE.

MD est un tableau unidimensionnel de nombres réels dont je vous laisse deviner la signification.

BSX prend une année au niveau 2 et un mois au niveau 1 de la pile et renvoie 1 si le mois est postérieur à février et si l'année est bissextile; il renvoie 0 dans tous les autres cas.

DAY prend une date comme argument et calcule le nombre de jours écoulés entre cette date et le 31 décembre de l'an zéro, en supposant que notre calendrier actuel était déjà en vigueur à cette époque.

DOW renvoie le jour de semaine d'une date (1 pour lundi, 2 pour mardi, ...).

DDAYS renvoie le nombre de jours écoulés entre deux dates. Ce nombre de jours est positif si la date niveau 2 de la pile est antérieure à celle au niveau 1. Il est négatif sinon.

DATE prend une date au niveau 2 de la pile, un nombre entier au niveau 1 et calcule la date obtenue en ajoutant ce nombre entier de jours à la date indiquée (Il correspond à la fonction DATE+ de la HP-48).

Convention d'écriture :

J'indique en face de chaque nom de programme l'état de la pile avant et après l'exécution du programme. Exemple :

DATE (2:date 1:jours → 1:date')

signifie que DATE a besoin d'une date ▬ niveau 2 de la pile, d'un nombre de jours ▬ niveau 1 et qu'il renvoie une nouvelle date au niveau 1.

Voici les listings de ces programmes :

MD

```
[ 0 31 59 90 120 151 181
212 243 273 304 334 366 ▮ ]
```

BSX (2:année 1:mois → 1:(0 ou 1))

```
« 2 > SWAP DUP 100
  MOD OVER 400 ▬ NOT
  OR SWAP 4 MOD NOT
  AND ▬
»
```

DAY (1:date → 1:jour)

```
« IP LAST FP 100 *
  IP LAST FP 10000 *
  ROT DUP2 BSX ROT 1 -
  DUP 365.25 * IP OVER
  400 / IP + SWAP 100
  / IP - + MD ROT GET
  + +
»
```

DOW (1:date → 1:jour_de_semaine)

```
« DAY 7 ▬ DUP 7
    IFTE «
```

DDAYS (2:date1 1:date2 → 1:date2-date1)

```
« DAY SWAP DAY - «
```

DATE (2:date 1:jour → 1:date+jour)

```
« SWAP DAY + DUP
  365.2425 / CEIL DUP
  1000000 / 12.309999
  + DAY ROT SWAP - ▬
  1
    WHILE GETI 4 PICK
    <
    REPEAT
    END 2 - SWAP OVER
  GET ROT SWAP - ROT
  ROT DUP2 BSX
    IF
    THEN ROT 1 -
    IF DUP NOT
    THEN DROP 1 - ▬
  OVER GETI ▬ ROT
  GET SWAP - DUP ▬ -
  NOT +
    ▬ ROT ROT
  END SWAP 1000000 /
  + SWAP 100 / +
»
```

Le répertoire TIME

Comme pour le répertoire DATES, je vais commencer par décrire un à un tous les objets avant de donner les listings.

Sauf indication contraire, toutes les heures sont à entrer sous la forme *hh.mmss*.

Remarque extrêmement importante

Les programmes qui suivent fonctionnent exclusivement sur des HP-28S version 2BB (rappel : pour obtenir la version de votre machine, entrez : HEX <ENTER> ▬ SYSEVAL). Si vous possédez une HP-28C version 1BB, remplacez TOUTES les occurrences de #11CAh SYSEVAL par #123Eh SYSEVAL. Hélas pour les possesseurs de HP-28C version 1CC, je ne connais pas l'adresse de la routine. Je vous conseille de vous mettre en mode hexadécimal (en tapant HEX) avant de commencer à pianoter sur vos claviers.

N1 contient la date (sous forme d'un nombre binaire) du dernier FADJ (voir plus loin).

TT est utilisé par le chronomètre.

MOIS contient les noms des mois de l'année.

JOURS contient les noms des jours de semaine.

TFACT contient la valeur du nombre TFACT dont j'ai déjà parlé.

DELTA et DAYS sont deux nombres permettant de calculer la valeur de delta dont j'ai déjà parlé.

→AFF a besoin d'un nombre entier au niveau 2 (ce nombre représente le numéro d'un jour) et d'une heure au niveau 1. Il affiche l'heure et la date correspondant à ses deux arguments.

F2 prend une heure comme argument et retourne une chaîne de caractères de la forme contenant l'heure avec un ▮ en tête si l'heure est inférieure à dix.

GTIME renvoie deux nombres : le premier est un nombre de jours et le second est l'heure actuelle.

→HJ est très similaire à GTIME. Il s'agit en fait de la principale partie de GTIME.

H→S transforme heure en chaîne de caractères.

DCORR permet de corriger l'heure. Il a besoin d'un nombre au niveau 1 de la pile. ce nombre est le nombre de secondes à ajouter (si positif) ou à retrancher (si négatif) à l'heure actuelle. Par exemple, -1.5 DCORR retarde l'horloge de une seconde et demie. DCORR ne réajuste pas TFACT.

FADJ prend un seul argument. Il s'agit du nombre de secondes d'avance que votre horloge pris depuis le dernier FADJ. Si votre horloge a pris du retard, entrez un nombre négatif dans la pile. Par exemple, si votre horloge pris treize secondes de retard depuis le dernier FADJ, tapez -13 FADJ.

Remarque :

Pour que FADJ fonctionne correctement, évitez d'employer DCORR entre deux FADJ. Si vous tenez quand même à utiliser DCORR, tapez 0 FADJ immédiatement après avoir exécuté DCORR.

CLADJ met l'heure à jour. Par exemple, s'il est 18 heures 13 minutes 43 secondes, tapez 18.1343 CLADJ.

DADJ met la date à jour. Par exemple, 12.251992 DADJ indique votre machine que nous sommes le 25 décembre 1992.

CLOCK affiche l'heure, le jour de semaine et la date à l'écran.

CONT efface l'écran et affiche continûment l'heure. Pour arrêter CONT, appuyez sur n'importe quelle touche autre que ON.

GO remet le chronomètre à zéro.

STOP indique le temps écoulé depuis le dernier GO.

Installation

Après avoir entré tous les programmes de cet article, revenez au répertoire TIME et effectuez les opérations suivantes :

1) Tapez mm.jjaaaa DADJ où mm.jjaaaa est la date actuelle

2) Tapez hh.mmss CLADJ où hh.mmss est l'heure actuelle

3) Vérifiez que votre horloge est précise à l'aide de CONT ou CLOCK.

4) Eventuellement utilisez DCORR pour faire de petites corrections sur l'heure.

5) Tapez 0 FADJ afin d'actualiser le dernier ajustement.

Une fois toutes ces opérations terminées, je vous conseille d'attendre quelques jours avant de réajuster votre horloge. Il est préférable d'utiliser FADJ au lieu de DCORR ou CLADJ, car cela permet au programme d'ajuster sa vitesse d'horloge.

Remarques :

Pour obtenir un bon facteur TFACT, il est nécessaire d'être très précis lors des ajustements d'heure et d'attendre plusieurs jours (voire plusieurs semaines) entre deux ajustements.

Voici les listings :

IT

```
0
```

N1

```
0
```

MOIS

```
{ " janvier "
" fevrier " " mars "
" avril " " mai "
" juin " " juillet "
" aout "
" septembre "
" octobre "
" novembre "
" decembre "
}
```

JOURS

```
{ "lundi" "mardi"
"mercredi" "jeudi"
"vendredi" "samedi"
"dimanche" }
```

TFACT

707788800

DELTA

0

DAYS

0

TT

0

→AFF (2:jour 1:heure →)

```
« CLLCD H→S 1 DISP
  DAYS + DATES
  1.000001 SWAP DATE
  DUP DOW TIME JOURS
  SWAP GET 2 DISP DUP
  F2 STD OVER FP 100 *
  IP →STR ROT MOIS
  SWAP GET + SWAP ▮ ▮
  SUB + ▮ DISP
»
```

F2 (1:nombre → 1:"nombre")

```
« DUP 9 ≤ "0" ""
  IFTE SWAP →STR + »
```

GTIME (→)

```
« ▮ 11CAh SYSEVAL
  B→R →HJ »
```

→HJ (1:nombre → 2:jour 1:heure)

```
« TFACT / DELTA + IP
  LAST FP 24 * →HMS »
```

H→S (1:heure → 1:"heure")

```
« ▮ FIX F2 DUP 1 2
  SUB " h " + OVER 4 5
  SUB + " m " + SWAP 6
  7 SUB + " s" + STD
»
```

DCORR (1:secondes →)

```
« 86400 / 'DELTA'
  STO+ »
```

FADJ (1:secondes →)

```
« 86400 / # 11CAh
  SYSEVAL B→R DUP2 N1
  - TFACT / ▮▮▮ - / 1
  + GTIME ROT 'TFACT'
  STO* 4 ROLL 24 *
  →HMS HMS- .000023
  HMS+ CLADJ GTIME
  DROP - 'DAYS' STO+
  'N1' STO
»
```

STOP (→)

```
« ▮ 11CAh SYSEVAL TT
  - B→R TFACT / 24 *
  →HMS DUP H→S 1 12
  SUB "." + SWAP F2 8
  9 SUB + " s" + 1
  DISP
»
```

GO (→)

```
« #11CAh SYSEVAL
  'TT' STO »
```

CLADJ (1:heure →)

```
« GTIME ROT SWAP
  HMS- .000015 HMS+
  HMS→ 24 / 'DELTA'
  STO+ DROP TIME
»
```

DADJ (1:date →)

```
« DATES DAY TIME
  GTIME DROP - 'DAYS'
  STO »
```

CLOCK (→)

```
« GTIME →AFF »
```

CONT (→)

```
« CLLCD 4 FIX
  DO #11CAh SYSEVAL
  B→R TFACT / DELTA +
  FP 24 ■ →HMS 1 DISP
  UNTIL KEY
  END DROP STD
»
```

Laurent Grand (516)

CHOC EN RETOUR

Dans mon article *Calcul formel sur HP-28S et HP-48SX* paru dans *JPC 77*, quelques erreurs ■ sont glissées :

- A la dernière ligne, au lieu de :

```
ROT OVER STO ■■■■ ■ faut lire :
ROT OVER STO ■ ORDER
```

- Le nom du programme est 'µ' (la lettre mu) et non la lettre 'u'.

- Enfin, les caractères ‹ et › doivent être remplacés par les délimiteurs de programmes « et ».

Philippe Heilbronn (233)

HP-48

G. Toublanc	Notes de lecture : *HP48 ■ Prépa*	*16*
L. Grand	Annuaire Téléphonique	*22*
G. Toublanc	Fast-orielles (Acte II)	*23*
G. Toublanc	Test de rapidité d'un programme	*26*
G. Toublanc	Critiques et réponses	*27*
G. Toublanc	Errare Toublancum est	*28*

HP 48 EN PREPA

notes de lecture

La HP-48 a acquis ses lettres de noblesse parmi les scientifiques et en particulier les matheux. Aux Etats Unis sont commercialisées au moins huit cartes de bibliothèques mathémathiques. La France tient bien son rang avec les ouvrages du milieu "prépa" :

- Ceux de J.M. Ferrard.
- Celui de P. Courbis et S. Lalande dont la vocation est de documenter les programmeurs en assembleur mais contient quelques programmes mathématiques intéressants.
- Et le tout nouveau *HP 48 en prépa* dont le titre annonce tout de suite la couleur et le sous-titre :

des solutions pour les concours!
programmes.astuces.assembleur.internals

précise les options prises par les auteurs qui se sont mis à trois pour nous concocter cet ouvrage de 398 pages.

Le livre est intéressant et ce genre d'entreprise est à encourager.

Le livre contient un bon de commande de la disquette des programmes que l'on peut se procurer pour la somme de 50 F + 10 F de frais de port. Cela nécessite évidemment la possession du kit de connexion HP/IBM PC

On peut répartir les chapitres en neuf parties :

Partie I 10 pages

introduction-prêt pour le prêt à programmer ?

C'est une préparation à l'utilisation de la HP-48. J'en retiendrai surtout la liste des frappes de touches pour obtenir les caractères spéciaux de la HP-48 et la structure des répertoires des programmes du livre parce que pour le reste il semble que le niveau des lecteurs auxquels s'adresse ce livre les fera passer rapidement les 7 autres pages.

Partie II 14 pages

Programmes de base
7 programmes répertoire HOME

Ce sont des programmes outils qui seront utiles non pas pour faire des mathématiques mais examiner des fichiers texte, assembler des chaînes de code ou des programmes ou bien faire l'inverse. Là je suis réticent au système employé pour les exemples d'utilisation des programmes. En effet par exemple pour lancer le programme LOOK on vous dit de faire la séquence de frappes :

```
[α] [α] [NEXT] [EVAL] [EVAL] [FH] [ENTER]
```

Alors que dans l'introduction on a expliqué comment obtenir les différents caractères. Cela présente l'inconvénient d'utiliser un pourcentage non négligeable de lignes et lorsque le nom, en clair, du programme se trouve 2 pages avant on est un peu désorienté. Ce qui est paradoxal c'est que lorsqu'un nom de sous-programme est à frapper dans un programme là on vous laisse agir comme un grand. Pour moi ce système est le point noir du livre qui par ailleurs a beaucoup de points positifs.

Ce programme LOOK permet de visualiser un texte rapidement. Cela est très pratique et remédie à la lenteur de l'éditeur de la machine. Dès ce programme j'ai remarqué que l'auteur de nombreux programmes ignore que le test :

```
IF 0 == THEN
```

peut être remplacé par

```
IF NOT THEN
```

ce qui est plus court et plus rapide.

De même l'instruction CASE rend la programmation d'une série de tests plus claire, plus courte et plus rapide.

FIND est proposé pour la recherche des chaînes.

PGM→ et →PGM permettent de décomposer ou de recomposer un programme. Dans chaque cas cela se réalise par deux SYSEVAL et c'est à utiliser avec précautions.

SYS permet de rappeler en mémoire un objet dont on donne l'adresse et opère aussi dans la mémoire cachée. Il utilise 2 SYSEVAL.

Enfin une application de SYS et →PGM est donnée et pour la réaliser il faut, oh horreur ! lire 8 lignes de frappes de touches qui se résument à ceci :

```
■ 18AD8h SYS ■ 60F9Bh SYS ■ →PGM
```

Pour assembler les chaînes de codes on utilise ASS qui est un programme en *user rpl* qui exploite l'astuce de travailler sur un objet graphique pour arriver à l'objet lui-même. Très bonne méthode qui évite les contraintes du procédé pour HP-28 et qui est si bonne

qu'elle n'est plus une nouveauté. Donc pas d'assembleur en assembleur.

Le décompilateur d'objet est cette fois en *system rpl* et assembleur. C'est l'équivalent de →ASC mais avec [redacted] octets de plus. Le listing du fichier source est fourni. Seule la partie en *system rpl* est commentée. En ce qui concerne la partie assembleur le code est aligné en face de chaque mnémonique mais il n'y [redacted] aucun label pour les sauts ou les boucles ce qui fait qu'il faut se livrer à un petit calcul qui n'est pas à la portée du débutant alors que le listing est proposé pour faire comprendre le fonctionnement du programme. Je doute que le but soit atteint. Les listes de codes pour tous les programmes en assembleur sont fournies sous forme de groupes de 3 caractères. Cela se mémorise peut être mieux entre la lecture et la frappe que des groupes de 4 ou 5 caractères moins gourmands en espace, c'est à voir.

Partie III 164 pages

L'arithmétique
16 programmes
Répertoire ARITHM

Les auteurs ont opté pour que les résultats soient souvent donnés sous forme de nombres rationnels. Aussi les programmes du répertoire ARITHM seront souvent utilisés comme sous-programmes.

On trouve au premier rang DECOMP et EDIVI qui ont fait leur première apparition dans JPC n° 62 de [redacted] 89 et pour la HP-28, sous la signature de votre serviteur. Lorsque j'ai acheté le livre j'ai eu un petit choc au coeur mais tout compte fait je suis content que ces programmes soient utilisés par un plus grand nombre de lycéens et d'étudiants. L'un des auteurs, Matthieu Cornillault, m'a promis de rétablir les choses lors d'une nouvelle édition. Je pense même que ma toute dernière version (JPC81) qui est plus performante et conviviale pourrait remplacer EDIVI ancien cru. Ici le listing du fichier source est proposé mais tout nu, sans commentaires ni labels, mais avec les adresses de désassemblage figurant en face des mnémoniques et permettant de localiser les sauts, boucles et sous routines.

On peut cependant regretter que les auteurs n'aient pas [redacted] la correction de faire figurer les noms des auteurs de nombreux programmes, laissant peut-être [redacted] nombreux débutants ignorant de ce qui se fait autre part l'illusion qu'ils étaient les auteurs de ces programmes... (note de [illegible])

Après ces 2 programmes pour la décomposition des nombres [redacted] facteurs premiers, les deux compères obligés sont PGCD et PPCM. Le premier a été réalisé en assembleur car il est souvent appelé par de nombreux programmes pour traiter les nombres rationnels. J'aurai l'occasion de revenir sur PGCD que je pensais être un thème largement épuisé or je me suis aperçu qu'il n'en est rien. Ici pas de fichier source pour PGCD seulement les codes. Je n'ai pu résister à la tentation de désassembler ce programme. J'ai découvert que l'algorithme qui est employé n'est pas celui classique d'Euclide, mais un algorithme plus récent découvert en 1961 par J.Stein donc plus de 2200 [redacted] après Euclide.

Suivent 12 programmes de calculs sur les fractions et un programme de redéfinition du clavier pour manipuler les fractions qui se présentent [redacted] forme de nombres complexes.

Calculs sur les polynômes
18 programmes
Répertoire ARITHM\POLY

Les polynômes sont repésentés par des listes de coefficients qui peuvent être des réels, des rationnels ou des valeurs litérales. Toutes sortes d'opérations peuvent être effectuées sur ces polynômes depuis celles de type arithmétique en passant par les PGCD et PPCM jusqu'aux opérations de dérivation et d'intégration. Ces programmes peuvent donc manipuler des polynômes de types différents et c'est ici que commence l'originalité du livre. La liste se termine par un programme de conversion des coefficients rationnels (...,...) en coefficients quelconques. Celui-ci fait appel 5 fois à SYSEVAL mais une adresse (#5E652h) ne se trouve pas dans la liste des internals. Cette lacune se reproduit dans d'autres programmes. Cela n'enlève rien à la valeur de l'ensemble mais fait buter le lecteur qui veut comprendre le programme et cherche la signification de certaines adresses dans la liste commentée des internals (voir ci-dessous).

Racines
2 programmes + des petits s/progr.
Répertoire ARITHM\POLY\RACINES

- Décomposition d'un nombre en ses diviseurs.
- Racines d'un polynôme, programme polyvalent apte à travailler sur des polynômes quelconques et dont les résultats peuvent être donnés sous forme rationnelle si le polynôme est à coefficients entiers.

Algorithmes
3 programmes
Répertoire ARITHM\ALGO

- Algorithme de Bézout bien connu des arithméticiens. Le résultat est envoyé dans une chaîne.

- Algorithme de Newton pour remplacer la fonction ROOT qui ne fonctionne qu'avec des réels.
- Et enfin un algorithme de tri de liste par insertion.

Développements limités à coefficients algébriques
23 programmes
Répertoire ARITHM\DL

Ici les DL se présentent sous forme de listes où les rationnels prennent forme de complexes. Donc de nouvelles possibilités par rapport à ce qui se fait ailleurs.

Un ensemble bien étoffé.

Pour ce répertoire les remarques faites pour le répertoire MATRICES concernant les tests de la présence d'arguments sont aussi valables.

calculs matriciels
15 programmes
répertoire ARITHM\MATRICES

Ces programmes travaillent sur des matrices de rationnels se présentant sous forme de complexes. On y trouvera les opérations classiques sur les matrices, et ce qui permet d'obtenir :

Co-matrice, polynôme caractéristique, déterminant.

Et pour finir la résolution d'un système de Cramer.

Je n'ai pas eu le temps d'analyser si l'algorithme de Leverrier qui est employé est l'original ou celui qui a été amélioré depuis. Aucune précision n'est donnée sur l'argument d'entrée et le résultat de ce sous-programme qu'un lecteur peut très bien désirer utiliser pour ses propres programmes. Donc celui-ci devra passer un certain temps avant de découvrir comment utiliser ce sous-programme pour lui-même. A mon avis les auteurs ont été un peu avares en explications concernant les programmes en général et que le lecteur, qui ne se contente pas d'utiliser seulement des outils mais veut comprendre et aller au-delà, se sent un peu frusté. Je sais qu'il faut bien limiter le volume d'un ouvrage pour des raisons économiques mais j'ai fait allusion, ici et là, à la possibilité de récupérer de la place.

Beaucoup de choses utiles dont une partie existe ailleurs mais ce qui fait la différence c'est la possibilité de travailler sur des rationnels.

Pour ce répertoire et ses sous répertoires il était possible de faire des économies d'octets, appréciables en mettant des programmes qui testent la présence d'arguments et délivreent un message d'erreur éventuel au lieu de faire faire ce travail par chacun des programmes. Ce qui a été fait, part d'une très bonne idée car elle constitue une aide en ligne appréciable lorsqu'on a oublié le nombre et la nature des arguments à fournir. Pour les répertoires MATRICES et DL on peut récupérer sans peine plus de 1200 octets. Je compatis pour les petits copains dont les machines ne sont pas bourrées de cartes (HP-48S etc..).

matrices algébriques
19 programmes
répertoire ARITHM\MATRICES\ALG

Les programmes permettent de faire des opérations sur des matrices pouvant contenir des éléments non évalués.

En plus d'utilitaires ont y trouvera les opérations des programmes du répertoire parent MATRICES.

applications des matrices
5 programmes
répertoire ARITHM\MATRICES\ALG\APPLI

- Jordanisation d'une matrice 3x3.
- Algorithme de Gauss, calcul du rang.
- Résolution symbolique d'un système.
- Equation des espaces propres d'une matrice. le résultat est donné sous forme litérale.
- et 3 sous-programmes utilisés par JORDAN.

Pour ceux qui n'ont pas la disquette, l'absence du checksum et du nombre d'octets des programmes est assez génant car par exemple pour JORDAN qui fait 2729 octets une erreur de frappe ne se décèlera pas immédiatement.

Outils
1 programme
Répertoire OUTILS

C'est essentiellement un programme de linéarisation de : cos(n*x) , sin(n*x) , cos^n(x) et sin^n(x).

Géométrie
2 programmes
Répertoire GEOMETRIE

- Un traceur d'enveloppes de droites.

- Développée d'une courbe paramétrée.

A propos de ces 2 programmes il est donné des exemples dont la présentation absorbe beaucoup de place et il était possible de mettre en tête de ces exemples les directives communes au lieu de les répéter inlassablement. Cette place, dans une nouvelle édition, pourrait être récupérée pour y mettre des compléments attendus par les lecteurs.

Equations différentielles
3 programmes
Répertoire EQUADIFF

Ces programmes doivent permettre de résoudre sous forme graphique des équations différentielles des premier et second ordres.

Cette deuxième partie repésente donc un ensemble de 108 programmes et sous-programmes.

Partie IV 40 pages

Sujets corrigés

Au travers de différentes épreuves pratiques de mathématiques des concours des grandes écoles on nous propose des solutions où la HP-48 apporte son concours soit directement ou par l'intermédiaire des programmes du livre. Evidemment la machine ne fait pas tout, il y a un travail théorique à faire avant l'utilisation de la machine. La possibilté des programmes de fournir des résultats sous forme de rationnels est mise à profit. Donc ce chapitre nous donne des applications intéressantes pour la clientèle "prépa".

J'avais signalé que l'un des auteurs semblait ignorer l'existence de l'instruction CASE aussi je ne puis résister à montrer l'intérêt de celle-ci avec l'aide d'un programme de ce chapitre et devant faciliter la solution numérique d'une partie de problème :

Voici le listing du programme en cause :

en 252.5 octets

```
« → n
  «
    IF ■ 2 MOD 0 ■
    THEN ■ 2 / DIVI
    ELSE
      IF n 3 MOD 0 ==
      THEN n 3 / DIVI
      ELSE
        IF n 5 MO 0 ==
        THEN n 5 / DIVI
        ELSE
          IF ■ 7 ■ 0 ==
          THEN n 7 / DIVI
          ELSE
            IF n 1 ==
            THEN 1
            ELSE 0
            END
          ■
        END
      END
    END
  »
»
```

Et maintenant la solution avec CASE :

en 207.5 octets

```
« → n
  «
    CASE n 2 MOD NOT THEN n 2 / DIVI END
         n 3 MOD NOT THEN n 3 / DIVI ■
         n 5 MOD NOT THEN n 5 / DIVI END
         ■ 7 MOD NOT THEN n 7 / DIVI ■
         n 1 ==      THEN 1          END
                          0
    END
  »
»
```

C'est plus clair et plus court.

Solution que l'on peut encore optimiser :

en 150 octets

```
«
  CASE DUP 2 MOD NOT THEN 2 END
       DUP 3 MOD NOT THEN 3 END
       DUP 5 MOD NOT THEN 5 END
       DUP 7 MOD NOT THEN 7 END
       DUP 1 ==      THEN 1 END
                          0
  END
  DUP 1 >
  IF
  THEN / DIVI
  ELSE SWAP DROP
  END
»
```

Le programme principal appelant ce sous-programme donne les résultats en :

avec la solution de 252.5 octets : 28 secondes
avec la solution de 207.5 octets : 19 secondes
avec la solution de 150 octets : 17 secondes

Ces comparaisons ne sont pas destinées à montrer ce qu'il faut faire dans un concours où on n'a pas le temps de paufiner dans une véritable course contre la montre.

Partie V 20 pages

trucs et astuces

On y donne [redacted] certain nombre de renseignements qui peuvent donner lieu à des astuces mais au sens très large du terme. Voici la liste :

- Remplacement des messages d'erreurs avec les *Array of String*.
- Menus. exemple pour réaliser un menu avec des libélés de différents types.
- Les ports de la HP-48SX. astuce pour rappeler les objets d'une carte (donc possibilité de piratage). C'est [redacted] moyen simple parmi toutes les méthodes possibles.
- Quelques précisions sur la commande PKT.
- Un programme PEEK.
- Les répertoires cachés. comment [redacted] créer.
- La commande WSLOG.
- Quelques informations sur le microprocesseur et le scanner.
- Descrition de la structure des objets HP-48 (plus succinte que dans l'ouvrage de P.C et S.L).

Partie VI 11 pages

programmation en assembleur

Ce chapitre peut se diviser en 2 parties :
- 4 pages d'explications sur la programmation en assembleur et le reste pour décrire les mnémoniques du processeur Saturn (HP 71/28/48). Les débutants trouverons certainement que c'est insuffisant d'autant que le meilleur moyen pour s'initier à l'assembleur c'est celui de la pédagogie par l'exemple. Or l'ouvrage ne contient que 2 fichiers sources non commentés et sans labels.

Partie VII 71 pages

internals de la HP 48

Les auteurs poussent un peu la plaisanterie en intitulant le chapitre : quelques internals de la HP 48. Dans ces 71 pages on trouvera classés par ordre croissant les adresses à la fois de ce qui est connu de l'utilisateur courant c'est-à-dire les commandes de la machine ainsi qu'un certain nombre d'objets tels 1, "A" et ce qui est caché c'est-à-dire programmes et routines internes sans oublier les objets non disponibles normalement tels que par exemples les system binary. Chaque adresse est documentée par une ligne. Du point de vue exploitation ces 71 pages [redacted] sont pas faciles à utiliser, il aurait fallu faire un classement par familles et séparer les commandes utilisateur du reste. En revanche pour le lecteur qui veut avoir une information sur une adresse utilisée dans un programme du livre c'est la solution de la liste unique qui convient. Il y aurait peut-être une possibilité d'améliorer les choses avec la disqette des programmes: il suffirait d'ajouter le fichier texte (en simple ASCII et donc lisible par tous) de la liste des internals, ce qui aurait l'avantage de permettre aux possesseurs de PC de faire plus rapidement des recherches avec un petit fichier batch et la commande FIND du DOS.

Partie VIII 34 pages

applications

Il ne s'agit pas d'applications des programmes mathématiques mais de 3 ensembles de programmes pour réaliser :

1 - Des bibliothèques. Il existe des programmes pour faire cela avec un PC mais ici on nous donne la possibilité de le faire avec une HP-48. C'est [redacted] très bonne idée que d'avoir fourni ce créateur de bibliothèques. Le mode d'emploi pour fabriquer est suffisant mais il manque quelques explications sur les propriétés des bibliothèques et en particulier il n'est pas signalé qu'elles ont une structure plate donc pas de sous-répertoires.

2 - des matrices de chaînes qui sont des objets se présentant sur la pile sous forme de *Array of String* et ne sont pas normalement accessibles à l'utilisateur. Elles sont utiles pour réaliser des ensembles de messages d'erreurs.

3 - des affichages à 5 ou 7 lignes.

L'ensemble des programmes est fait pour réaliser une bibliothèque. A l'usage cela se révèle très pratique. Je connaissais des programmes qui font ces affichages étendus mais l'affichage n'est pas permanent. On peut passer indifféremment de 5 à 7 niveaux. Pour 5 niveaux il y [redacted] des fonctionnalités supplémentaires.

Partie IX 21 pages

annexe : programmes et sous-programmes

Cette liste récapitulative des programmes donne par répertoire le nom des programmes et indique si le programme est autonome ou s'il fait appel à d'autres programmes du même répertoire ou de répertoires différents. Ceci est une bonne chose car cela permet d'avoir l'assurance que si l'on extrait une partie des

programmes il n'y aura pas de plantage parce qu'il manque un sous-programme appelé par le programme principal. Cette façon d'opérer sera particulièrement utile pour les possesseurs de HP-48S qui [redacted] pourront pas loger l'ensemble des programmes mathématiques du livre.

Pour l'avenir :

Lors d'une nouvelle édition l'ouvrage pourrait être encore plus apprécié en lui apportant les améliorations suivantes :
- Systématiquement indiquer le checksum et le nombre d'octets de tout programme ou objet. Cela est indispensable pour ceux qui ne possèdent pas la disquette de programmes et cela permettrait à tous d'identifier les versions lors d'une actualisation des programmes.
- Indiquer le temps d'exécution pour obtenir les résultats des exemples.
- Sur la disquette fournir le fichier texte des internals pour faciliter les recherches à l'aide d'un fichier batch et de la commande FIND du DOS.
- Abandonner la méthode des frappes de touches mais donner simplement le [redacted] du programme à appeler pour obtenir le résultat des exemples.
- Donner les fichiers sources de tout programme en assembleur avec les labels pour les sauts ou les appels de sous routines. Des commentaires seraient les bienvenus.
- Partitionner la liste des internals par familles et quelques informations sur le microprocesseur et la compléter avec les internals utilisés dans les programmes mais non documentés.
- améliorer la rapidité des programmes de calcul formel :
-en faisant appel à des programmes écrits en *system rpl* ou en assembleur lorsque cela est possible.
-en évitant que plusieurs sous-programmes fassent des vérifications d'argument alors que cela ne doit être fait qu'une fois par le programme principal.
-en optimisant les programmes par une meilleure utilisation des instructions de la HP-48 par exemple utilisation de l'instruction CASE dans une suite de tests.
- Faire un sous-programme d'aide en ligne appelable par les nombreux programmes qui utilisent le même type d'aide en ligne au lieu de l'intégrer dans chacun des programmes d'où un encombrement mémoire excessif.

Réaliser un ouvrage en équipe présente à la fois des avantages et des inconvénients :

- Avantages : utilisation des compétences de chacun et répartition du travail.
- Inconvénients : il faut assurer [redacted] coordination parfaite et rendre l'ensemble homogène.

L'ouvrage a bénéficié des avantages mais parfois aussi des inconvénients. Mais n'oublions pas que les auteurs sont des étudiants qui ont donné beaucoup d'eux-mêmes et ont disposé d'un temps limité par leurs études.

En conclusion nous pouvons dire que cet ouvrage est [redacted] oeuvre utile et intéressante qui mérite de faire [redacted] chemin. Les auteurs peuvent encore, avec un peu de recul, rendre ce livre tout à fait indispensable. Nous devons être reconnaissants [redacted] auteurs d'avoir sacrifié leurs loisirs pour réaliser ce travail important et nous ne pouvons souhaiter qu'une chose : qu'ils continuent.

Je n'ai pas la prétention d'avoir tout vu de cet ouvrage ni tout dit. Il faut beaucoup plus de temps que je n'en ai disposé mais je pense que vous pouvez maintenant avoir une petite idée de son contenu. Il aurait fallu parler de façon précise de la rapidité d'exécution des programmes et en particulier de ceux qui sont très polyvalents car pouvant travailler avec des données de types différents mais dont la contre-partie est d'être plus lents que les programmes plus spécialisés. Il aurait fallu aussi tester les programmes avec des arguments plus compliqués que ceux des exemples du livre. A vous de le faire et de nous [redacted] parler.

Bonne lecture.

Guy Toublanc (276)

HP 48 en prépa
par Matthieu Cornillault
Marc de Courville
Emmanuel Lesueur
Editions Dunod
165 francs

Note : Un erratum de 2 pages accompagne maintenant le livre. Il concerne des erreurs dans les listings de programmes. Cet erratum doit être disponible auprès des libraires pour les premiers acheteurs du livre, ce qui est honnête alors que pour le livre de P.C et S.L, il faut acheter la 2ième édition pour avoir [redacted] texte épuré, [redacted] qui est en régression par rapport [redacted] ce qui avait été fait pour leur ouvrage sur la HP-28.

ANNUAIRE TELEPHONIQUE

TEL est un petit annuaire. Il vous permet de stocker, de rechercher et de supprimer des adresses, ou plus généralement des enregistrements. Il a été écrit pour la première fois en 1989 pour une HP-28S. Comme je possède actuellement une HP-48, et que je n'ai plus de HP-28, je vous présente les programmes pour la HP-48. Les possesseurs de HP-28 n'ont qu'à changer la commande LASTARG en un LAST et supprimer les commandes 3 FREEZE du programme AFF.

Tous les enregistrements sont stockés dans la variable BOTIN. La variable BOTIN doit contenir une chaîne de caractères vide avant la création du premier enregistrement.

Remarque importante :

La dernière phrase du premier paragraphe vous a sans doute fait deviner qu'il est nécessaire que LASTARG (ou LAST pour les HP-28) soit valide.

Mode d'emploi

CREER: (string -->)

Permet d'ajouter un nouvel enregistrement. string doit être composé d'au moins deux lignes. La première ligne contiendra le nom de la personne. Lors de la recherche, le critère de choix s'effectuera sur la première ligne de chaque enregistrement.

Exemple :

```
"BOLOMEY JACQUES
Avenue des Canaris 12
11223 Fantômeville
tel 12.34.56.78" CREER
```

J'ai écrit le nom en majuscules uniquement pour faciliter une recherche ultérieure.

NOTEL : (string -->) ou (expr -->)

Cherche le premier enregistrement dont la première ligne commence par string (ou expr).

Exemple :

```
'BOL' NOTEL
```

Attention : NOTEL tient compte des minuscules et des majuscules.

SUIVANT : (-->)

Cherche le prochain enregistrement correspondant à l'argument du dernier NOTEL.

Attention : LASTARG contient une partie de BOTIN et le nom à rechercher. Il ne faut donc utiliser SUIVANT qu'immédiatement après SUIVANT ou NOTEL.

TOUS : (-->)

Il s'agit d'un raccourci pour "" NOTEL.

TOUS permet ainsi de visualiser séquentiellement tous les enregistrements en exécutant plusieurs fois le programme SUIVANT après TOUS.

OTER : (string -->) ou (expr -->)

Supprime l'enregistrement dont la première ligne est *exactement* string ou expr.

Exemple :

```
"BOLOMEY JACQUES" OTER
```

Ici, l'espace entre BOLOMEY et JACQUES nous contraint à utiliser une chaîne de caractères comme argument.

Et maintenant, voici les programmes :

BOTIN

```
""
```

AFF

```
« 3 FREEZE CLLCD
DISP
»
```

OTER

```
« →STR 2 OVER SIZE
1 - SUB DUP BOTIN 1
CHR ROT + 10 CHR +
POS
  IF
  THEN BOTIN 1
       LASTARG 1 - SUB
       LASTARG + 1 + BOTIN
       SIZE [illegible] DUP 1 CHR
       POS
       IF
       THEN LASTARG
```

```
            OVER SIZE SUB +
        ELSE DROP
        END
        'BOTIN' STO
        DROP
    ELSE "
Non repertorie" +
        AFF
    END
»
```

CREER

```
« 10 CHR + LASTARG
1 CHR ROT + DUP ROT
POS
  IF
  THEN LASTARG OVER
       1 ROT SUB BOTIN
       SWAP POS
       IF
       THEN DROP DUP
            10 CHR POS 1 SWAP
            SUB
            "Deja repertorie" +
            AFF
       ELSE BOTIN +
           'BOTIN' STO DROP
       END
  ELSE DROP
       "Non conforme" +
       AFF
  ■
»
```

NOTEL

```
« →STR 2 OVER SIZE
1 - SUB BOTIN 1 CHR
ROT + DROP2 SUIVANT
»
```

TOUS

```
« "" NOTEL
»
```

SUIVANT

```
« LASTARG DUP2 POS
  IF
  THEN 1 LASTARG +
       ROT SIZE LASTARG
       ROT ROT ■ DUP 1
       CHR POS
       IF
       THEN LASTARG 1
            SWAP 1 - SUB
            LASTARG + OVER SIZE
            SUB
       ELSE ""
       END ROT ROT AFF
  END DROP2
»
```

J'espère que vous saurez apprécier la simplicité et l'utilité de cet ensemble de programmes.

Laurent Grand (516)

FAST-ORIELLES

Acte II

Notre ami Laurent Grand nous a gratifié de FFACT, (JPC-81), programme de calcul de factorielles en multi-précision dont la rapidité d'exécution est ■ commune mesure avec les programmes en User RPL. Il est même 10 fois plus rapide que BFACT de l'ouvrage de P.Courbis et S.Lalande, ce programme étant partiellement en assembleur. Laurent laissait présager des améliorations possibles par un autre programmeur. Je me suis mis sur la liste, et parmi différentes versions de programmes j'en ai retenu deux qui sont un compromis entre encombrement et rapidité. L'un des programmes admet des arguments de 5 chiffres donc éventuellement utilisable par une HP-48SX avec une extension mémoire, l'autre ■ limite à 4 chiffres donc destiné aux HP-48S ou HP-48SX sans extension mémoire. En effet, grâce à la formule de Stirling on peut calculer que pour 10000 ! il faudra 35665 octets de mémoire pour la seule chaîne du résultat, ce qui dépasse largement la capacité mémoire des HP-48 nues. D'autre part si l'on estime que des arguments inférieurs à 1E4 sont largement suffisants alors la version pour HP-48SX doit être abandonnée car 11% moins rapide que sa soeur. Je ne présente pas les versions plus rapides car plus gourmandes en octets.

Quelques remarques concernant le programme FFACT de Laurent :

- ■ réel entier est demandé comme argument mais aucune vérification n'est faite sur la nature de celui-ci ce qui cause quelques problèmes car pour des arguments inférieurs ou égaux à 14 c'est FACT qui fait le calcul et il y ■ divergence entre elle et FFACT au niveau des résultats et des messages d'erreurs pour les arguments de types suivants :

réel négatif, réel non entier, complexe

- le mode symbolique conduit à l'erreur

- il n'y a pas de limite d'argument:

1048577 FFACT renvoie 1048577d
alors que le programme est conçu pour n < 100000

Dans mes programmes j'ai inclu les vérifications nécessaires des arguments ce qui coûte 35.5 octets.

Voyons les résultats de mes améliorations :

pour calculer 1000 ! il faut 82 secondes avec la version lente et 74 secondes avec sa soeur contre 202 secondes pour FFACT de Laurent.

Si je n'inclus pas les vérifications d'arguments mes versions occupent 228.5 et 230 octets contre 378.5 pour celle de Laurent.

Ce gain d'octets a été obtenu :

- en utilisant des routines en *System RPL* qui remplacent une partie du code en assembleur
- en reprogrammant la formule de Stirling et en négligeant la partie résiduelle 1/12n qui n'est pas utile pour le calcul du nombre de chiffres de n !.
- en regroupant (dans la partie assembleur) les 2 boucles l3 et l4 en une seule.

Le gain de rapidité a été obtenu :

- par une utilisation optimale des registres et une réduction du nombre de transferts de données entre registres et mémoire: n ! est découpée en tranches de 11 à 12 chiffres au lieu de 5 pour la version de Laurent
- je n'utilise pas la routine *multiply* mais une copie légèrement simplifiée. Voici le listing de *multiply* :

```
DB9B  b=0     w
DB9E  sb=0
      csrb
      ?sb=0
      goyes   DBAC
      b=b+a   w
DBAC  a=a+a   w
      ?c#0    a       * modif
                      * ?c#0   a
      goyes   DB9E
      a=b     w       * supprimé
      c=a     w       * supprimé
      rtncc
```

ce qui me permet de ne pas détruire le registre C et de pouvoir utiliser son contenu pour la suite:
- utilisation de la pile des retours, quand cela est possible, pour les sauvegardes car plus rapide et moins gourmand en octets.
- quand cela a été possible le chargement d'une même constante ou la sauvegarde d'une donnée d'une façon répétitive dans une boucle ont été évités.

Voici le fichier source de la version de FFACT pour HP-48S. Les modifications pour la version HP-48SX sont indiquées. Aux modifications près, la programmation suit celle de Laurent dont presque la totalité des commentaires s'applique.

FFACT (version HP-48S)

```
RPL

::
CK1&Dispatch                ( vérifie qu'il y a  )
real                        ( 1 réel sur la pile )
::
DUP                         ( vérificaions de la )
%SGN                        ( validité de        )
%0<                         ( l'argument :       )
OVER                        ( entier positif     a
%FP                         ( et inférieur a la  )
%0<>                        ( limite             )
OR                          (                    )
OVER                        (                    )
DOREAL                      (                    )

ASSEMBLE

        nibhex    4000000000000010
                        a 10000

                  a nibhex    5000000000000010
                        a 100000 pour hp48sx
RPL

%>=                         (                    )
ORNOT                       (                    )
?SKIP                       (                    a
SETSIZEERR                  ( mauvaise valeur    )
DUP
%15
%<
case
::
%NFACT                      ( FACT si n < 15     )
a%>$                        ( n ! en chaine      a
;
TOTEMPOB                    ( sinon calcul du    )
DUPDUP                      ( nombre de chiffres )
%e                          ( formule de Stirling )
%/                          ( simplifiée         )
```

```
%LOG
%*
OVERDUP
%+
%PI
%*
%SQRT
%LOG
%+
%CEIL
COERCE                    ( nombre de chiffres)
NULL$TEMP                 ( création de la     )
OVER                      ( chaine devant      ]
#2*                       ( recevoir n !       ]
EXPAND                    (                    )
SWAPROT
COERCE                    ( n en system bin.   ]

CODE

        gosbvl    =POP2#
        acex      ■
        rstk=c              ■ longueur de n!
        c=0       ■
        c=a       a
        r2=c                ■ n hexa
        gosbvl    =SAVPTR
        a=dat1    a
        d1=a
        d1=d1+    10
        ad1ex
        r1=a                * adresse début n!
        d1=a
        c=0       a
        r0=c
        c=c+1     ■
        p=        11        * 10 pour hp48sx
        dat1=c    wp

loop
        c=r1
        d0=c
        c=r0
        d=c       ■
        c=0       a
l1      rstk=c
        setdec
        a=0       ■
        a=dat0    wp
        c=r2
        b=0       ■         * routine multiply
mult1   sb=0                ■ intégrée ici
        csrb
        ?sb=0
        goyes     mult2
        b=b+a     ■
mult2   a=a+a     ■
        ?c#0      a         ■ car n < 1E5
        goyes     mult1
        c=rstk
        c=c+b     w
        dat0=c    wp
        d0=d0+    12        ■ 11 pour hp48sx
        c=0       wp
        cslc
        cslc
        cslc
        cslc                                  * cslc à ajouter
                                              ■ pour hp48sx
        sethex
        d=d-1     a
        gonc      l1
        ?c=0      a
        goyes     l2
        dat0=c    ■
        a=r0
        a=a+1     a
        r0=a
l2      c=r2
        c=c-1     ■
        r2=c
        ?c#0      a
        goyes     loop

        p=        0         ■ pour la sortie
        c=rstk
        c=c-1     ■
        b=c       a
        csrb
        d=c       a
        c=r1
        c=c+b     ■
        d0=c
        rstk=c
loop2   a=dat0    1
        c=dat1    1
        dat1=a    1
        dat0=c    1
        d1=d1+    1
        d0=d0-    1
        d=d-1     a
        gonc      loop2
        c=rstk
        d1=c
        c=c+b     ■
        d0=c
        lchex     30      * chargé avant la boucle
l3      c=dat1    1       ■ le transfert ▬ ascii
        dat0=c    2       * ne ▬▬▬▬ qu'ici
        d0=d0-    2
        d1=d1-    1
        b=b-1     a
        gonc      l3
        govlng    =GETPTRLOOP
```

```
ENDCODE

;
;
```

Vous trouverez la liste des codes dans la rubrique *le coin des codes* pour les deux versions.

Remarque: ci-dessus j'ai fait allusion à d'autres de mes programmes qui sont plus rapides mais plus encombrants. En effet en employant les nouvelles instructions telles que : `a=r0.f csrb.f` etc..

on peut améliorer la rapidité mais en sacrifiant des octets.

Merci à Laurent pour avoir montré le chemin.

Guy Toublanc (276)

BON TEMPS ET BONS PROGRAMMES

Vous êtes riches de plusieurs programmes qui remplissent le même rôle et vous désirez garder le meilleur. Pour faire ce choix, deux critères sont à examiner : l'encombrement mémoire et la rapidité.

Je vous propose [redacted] petit programme permettant de tester la rapidité. Celui-ci est en *RPL* pour en réduire l'encombrement. Pour les débutants en RPL je signale ici l'usage de variables locales sans nom (gain d'octets et de rapidité).

TIMED

syntaxe :

niveaux 2 3 ... : les arguments éventuels
niveau 1 : le programme ou son nom

résultat : le temps d'exécution en sec.

Voici le fichier source :

```
RPL                 ■ mode rpl

::                  ■ début programme
DUPTYPEIDNT?        ■ niveau 1 : nom global ?
IT XEQRCL           ■ si oui RCL
xMEM                * MEM pour mettre de l'ordre
                    ■ dans la mémoire
WORDSIZE            ■ rappel de la longueur des
                    ■ entiers binaires
SIXTYFOUR           * system binary <64d>
dostws              * met à 64 la longueur des
                    ■ entiers binaires
{ NULLLAM NULLLAM }* crée 2 variables locales [redacted]
                    ■ [redacted] :
                    ■ niveau 2 -> variable 2
                    ■ niveau 1 -> variable 1
BIND                ■ crée l'environnement
CLKTICKS            ■ TICKS ( entier binaire )
                    * temps de départ
2PUTLAM             * temps de départ -> variable 2
EVAL                ■ exécute le programme à tester
CLKTICKS            ■ temps après l'exécution
2GETLAM             ■ variable [redacted] -> temps de départ
x-                  ■ différence entre les 2 temps
#>%                 ■ convertit la diff. en réel


ASSEMBLE            ■ mode assembleur


      con(5) #efee  ■ réel 8192 pour convertir [redacted]
                    ■ secondes
RPL                 ■ mode rpl

%/                  * division de [redacted] réels
                    ■ différence en secondes
DOREAL              ■ prologue réel

ASSEMBLE            ■ mode assembleur

      nibhex 7990000000007530
                    * réel 0.00375 pour tenir
                    ■ compte du temps d'exécution
                    ■ de ce qui n'est pas le pro-
                    ■ gramme proprement à tester
RPL                 ■ mode rpl

%-                  ■ différence entre 2 réels [redacted]
                    ■ temps - 0.00375 [redacted]
1GETLAM             ■ variable 1 -> wordsize départ
dostws              ■ stocke wordsize
ABND                ■ abandonne l'environnement des
                    * variables temporaires
;                   * fin programme
```

J'ai choisi le facteur d'ajustement de 0.00373 seconde pour trouver des résultats sensiblements égaux à ceux fournis par le programme *TIMED* de W.C. Wickes que le club anglais utilise pour les programmes à optimiser.

Vous trouverez la liste des codes dans *le coin des codes*.

Bonne chasse aux millisecondes.

Guy Toublanc (276)

Y A QU'A ET RECIDIVE

Lorsque j'ai créé la rubrique *Assembleur, les outils du début* nous étions dans la situation suivante :

- Le club venait de reprendre ses activités après de longs moments de torpeur. Nous avions ainsi pris beaucoup de retard vis à vis de la HP-48. Entre temps des gens actifs hors club ont fait un bon bout de chemin et ont créé leurs bibliothèques.

- P. Courbis et S. Lalande ont sorti leur ouvrage *voyage au centre de la HP-48* fournissant ainsi un bon ensemble d'explications et de programmes.

Alors fallait-il renoncer à publier des choses qui existaient ailleurs mais pouvaient être améliorées ou même présentées avec un minimum d'explications ?

En effet les auteurs de bibliothèques ne dévoilent pas leur programmation qu'ils gardent secrète.

Concernant l'ouvrage cité, s'il constitue encore le livre référence indispensable par la très riche documentation et les programmes qu'il fournit, il souffre pas moins de quelques faiblesses :

Pratiquement, il ignore la programmation *system Rpl* et de ce point de vue aurait besoin d'être actualisé. D'autre part certains programmes sont pas optimisés et comportent même des petites maladresses (sans aucun effet sur la fiabilité du résultat). La deuxième édition (sortie des presses en mars 92) que je n'ai pu voir qu'à Paris (on vend encore la première édition au *Furet du nord*, à Lille, qui se targue d'être la plus grande librairie du monde !), n'a que des corrections mineures du point de vue du fond et l'ajout d'explications sur les *linked arrays*.

J'ajouterai qu'en général ce qui se crée à Paris hors club PPC n'atteint pas les provinciaux.

Pour toutes ces raisons j'avais donc pensé que je pouvais lancer la rubrique assembleur comme je l'ai fait en faisant profiter les lecteurs de *JPC* de programmes actualisés, optimisés (on peut faire certainement encore mieux) et avec un minimum de commentaires. Mon objectif étant d'inciter le plus grand nombre possible de programmeurs à élargir cette rubrique. D'autre part l'existence de ces outils doit permettre à tous d'utiliser les mêmes programmes, portant les mêmes noms, avec la même syntaxe et ne pas être obligé lors d'une création de programme dans JPC de renvoyer à l'ouvrage X ou à la bibliothèque Y de Z.

Y A QU'A

Si je réagis ainsi c'est qu'une réflexion, mettant en cause l'effort que j'ai fourni, m'est parvenue. Or les colonnes de JPC sont ouvertes à tous (même non adhérents) et ne demandent qu'à recevoir les travaux des auteurs les plus divers sur les sujets les plus divers pouvant profiter à tous. Mais je crois que pour certains leur travail est parfait parce qu'inexistant et qu'ils ont le beau rôle en critiquant ceux qui ont plus de courage et sont moins passifs.

RECIDIVE

Hors le programme CIRCLE, dont programmation est plus performante que celle de l'auteur de CIRCLE du livre de P.C et S.L, j'ai eu la réserve de ne pas établir de comparaison de programmes avec ceux de l'ouvrage cité, laissant ce soin aux lecteurs attentifs et soucieux du détail.

J'avais promis de donner une version en assembleur du programme REV réalisé en *system rpl* (JPC 80). Aussi je profite de cet article pour présenter ce programme et mettre les points sur les "i" du personnage blasé.

Rappel: REV inverse l'ordre des caractères d'une chaîne.

```
*****************************************************
*        REVERSE          *        REV               *
*   (Courbis Lalande)     *                          *
*****************************************************
*      64.5 octets        *    60.5 octets           *
*****************************************************
*  accepte tout type      *     n'accepte            *
*  d'argument qu'il       *   qu'un argument         *
* transforme en chaîne    *       chaîne             *
*                         ***************************
*                         *  pour 10 caractères     *
*                         *  5 fois plus rapide     *
*                         *  pour 260 caractères    *
*                         *  2.2 fois plus rapide   *
*****************************************************
*      92 octets          *                          *
*****************************************************
*  doit appeler CHK       *                          *
*   ( 207 octets )        *                          *
*ainsi REVERSE n'accepte*                            *
* qu'un argument chaîne *                            *
*                         *  pour 10 caractères     *
*                         * 2.3 fois plus rapide     *
*                         *  pour 260 caractères    *
*                         * 1.45 fois plus rapide   *
*****************************************************
```

Comme il est probable que ces programmes seront utilisés en général avec de petites chaînes mais souvent (exemple: un programme de désassemblage) je vous laisse tirer les conclusions.

Je ne vais pas analyser, point par point, les différence entre REVERSE et REV mais lister la boucle qui fait l'essentiel du travail.

```
            REV                    REVERSE
      loop
181         d0=d0-  2     ■
14F         c=dat1  b     *          a=dat1  b
14A         a=dat0  b     ■          c=dat0  b
149         dat1=a  b     ■          dat1=c  ■
14C         dat0=c  b     *          dat0=a  ■
171         d1=d1+  2     *          d1=d1+  2
                          *          d0=d0-  2
                          *
                          *  133     ad1ex
                          ■  131     d1=a
                          ■  136     cd0ex
                          ■  134     d0=c
                          ■
CD    decr  b=b-1   a     *  8BA     ?c>=a   a
5BE         gonc    loop  *  FD      goyes   loop
```

Le principe de ces programmes est des plus simples et laisse peu de marge pour innover mais je pense que même dans ce cas REV mérite les colonnes de JPC pour ses différences et que les lecteurs que le sujet intéresse pourront aller plus loin dans l'analyse du fichier source ci-dessous.

REV

```
RPL

::
CK1&Dispatch                  ( vérifie s'il y a )
str                           ( bien 1 chaîne ■ )
                              ( sur la pile )
::
DUPLEN$                       ( niveau 2 : chaîne )
                              ( niveau 1 : system )
                              ( bin.=long. chaîne )
CODE

      gosbvl  =POP#           * A(A) := nombre car.
                              * DROP 1 niveau
      gosbvl  =SAVPTR         * sauve les pointeurs
      b=a     ■               ■ nombre caractères
      a=dat1  a
      d1=a
      d1=d1+  10              * ■ les données de la
                              * chaîne
      cd1ex
      d1=c
      c=c+b   a
      c=c+b   ■
      d0=c                    * ■ après la chaîne
      bsrb.f  a               ■ pour l'échange
                              ■ symétrique des car.
      gonc    decr            * saut -> compteur
loop
      d0=d0-  2               ■ actualise position
                              ■ en fin de chaîne
      c=dat1  b               ■ caract. début
      a=dat0  b               * échange
      dat1=a  b               ■ symétrique
      dat0=c  b               ■ des caractères
      d1=d1+  2               ■ actualise position
                              * ■■ début de chaîne
decr  b=b-1   a               * compteur d'échanges
      gonc    loop            ■ boucle continue si
                              ■ compteur >= ■
      govlng  =GETPTRLOOP     ( restaure les )
                              ( pointeurs ■ retour)
                              ( au RPL ■ )

ENDCODE

;
;
```

Vous trouverez la liste des codes de REV dans *le coin des codes*.

Conclusion :

Si beaucoup de programmeurs ont une dette envers P. Courbis et S. Lalande, qui ont accompli un très gros travail en plus de leurs études et cela aux dépens de leur vacances certainement, ce n'est pas une raison pour ne plus rien faire. Le problème actuel de *JPC*, c'est la passivité de beaucoup de ses lecteurs, adhérents ou non du club.

Guy Toublanc (276)

ERRARE TOUBLANCUM EST

Parmi mes articles parus depuis le n° 78, se sont glissées quelques erreurs ou maladresses :

JPC 78 page 17. Ne pas tenir compte de :

...il utilise le champ A du registre R4 pour les sauvegardes alors que ce champ est réservé pour gérer les interruptions.

Cela est vrai pour la HP-71 et j'avais retrouvé cette affirmation chez un auteur américain. Or, en désassemblant des routines de la Rom HP-48, je suis aperçu qu'il n'en était rien, ce qui est très bonne chose pour le programmeur.

JPC 80 page 13. Dans le listing du fichier source de PIL je suis passé, inutilement, deux fois en mode assembleur. Remplacer :

```
ASSEMBLE            |
     con(5) #1cc6b  |      par
RPL                 |
```

de même :

```
ASSEMBLE            |
     con(5) #2a2c9  |      par      %1
RPL                 |
```

Cela ne change rien résultat après compilation et donc la liste des codes est identique.

JPC 80 page 14. Le nota bene :

le compilateur n'accepte les commentaires que pour les parties non RPL.

est à remplacer par :

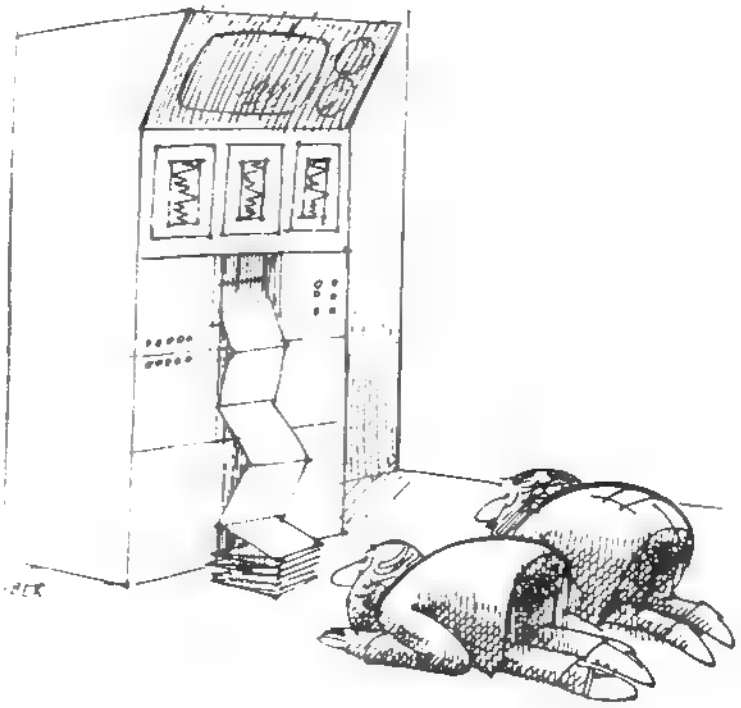
le compilateur accepte, pour les parties RPL, les commentaires entre parenthèses.

D'autre part en relisant mes articles dans JPC j'ai la désagréable surprise de voir ici et là des fautes orthographiques honteuses. Ces étourderies sont souvent le résultat d'un manque de temps pour relire le texte ou de modifications partielles de phrases pour rendre le formatage par lignes de 50 caractères, moins disgracieux :
(grands espaces entre les mots).

J'ai la maigre consolation de penser que ceci ne rend pas les articles incompréhensibles et qu'il vaut mieux cela que des pages blanches qui seront toujours orthographiquement irréprochables.

Guy Toublanc (276)

Avant PPC PARIS...



Après....

Dieu n'existe pas.

Alors, qui nous a allumé ?

HP-95

J. Belin Du Saturn au 8086 (Acte III) *32*

DU SATURN AU [redacted]

Acte III

Cet article constitue la troisième partie d'une série d'articles consacrée à l'assembleur sur le HP95, destiné plus particulièrement aux anciens programmeurs sur HP-71.

Après avoir discuté de la structure des programmes MS-DOS autour d'un exemple standard, nous allons aujourd'hui écrire un "vrai" programme faisant appel à différentes fonctions standard du BIOS de l'IBM PC (affichage, accès au clavier) ou d'autres spécifiques au HP95.

Les anciens membres du club se souviennent peut-être que le tout premier programme HP-71 paru dans JPC (dans le numéro 13, pour être précis) permettait de contrôler interactivement le contraste de l'affichage. Ce programme était écrit en Basic et utilisait la commande CONTRAST du HP-71. Le HP95 permet bien de règler le contraste à partir du SETUP quand le System Manager est activé, mais cela n'est pas possible à partir de la ligne de commande du DOS, car aucune commande n'est prévue à cet effet. Bien que certains utilitaires, tel 95.COM de Richard Harvey permettent de pallier ce problème, nous allons écrire la version en assembleur du programme Basic précédemment cité.

Son principe sera de contrôler interactivement le contraste à l'aide des flèches verticales, puis le fixer grâce à la touche ENTER, ou abandonner tout changement grâce à la touche ESC. De plus, nous allons afficher les valeurs initiales et courantes du contraste. Bien sûr, ce programme utilisant des fonctions propres au HP95, il faudra en interdire l'exécution sur un PC classique.

Organigramme Général du programme

```
Détection de fonctionnement sur HP95
Si pas HP95
  Sortie du programme
Lecture du contraste initial
début de boucle
  Affichage
  Lecture du clavier
  Si touche ESC
    Sortie [redacted] restauration du contraste initial
  Si touche ENTER
    Sortie avec conservation du nouveau contraste
  Si touche UP
    Augmentation de contraste d'un point
  Si touche [redacted]
    Diminution du contraste d'un point
Fin de boucle
```

Ceci n'est qu'un organigrame simplifié, vous permettant de comprendre le fonctionnement du programme. La version définitive sera légèrement différente, mais nous verrons cela plus tard.

En attendant, nous allons étudier différentes fonctions qui nous seront utiles : la détection du HP95, l'affichage, la lecture du clavier et enfin le contôle du contraste.

Détection du HP95

Ayant déjà traité ce sujet dans *JPC 76*, je ne ferais que rapeller ici que nous utiliserons la fonction 4d, [redacted] fonction d4 de l'interruption 15, qui retourne des valeurs prédéfinies pour le HP95.

Affichage

Afin de simplifier ce programme, nous utiliserons une méthode "brute", consistant à réafficher la totalité du texte, à l'aide de la fonction 9 du DOS (voir l'acte II). La mise à jour des valeurs affichées se fera en plaçant directement celles-ci (après conversion en caractères ASCII) dans la chaîne à afficher.

Mais, afin d'avoir un affichage stable (qui le réaffiche à la même position de l'écran et non sur les lignes suivant le texte) nous devrons contrôler la position du texte affiché sur l'ecran. Pour cela, nous utiliserons les fonctiones du BIOS permettant de lire la position de curseur, puis de le repositionner. Une fois que nous avons affiché notre texte, le curseur se trouve sur la première colonne de la ligne suivant la fin du texte. Si nous soustrayons la longueur du texte au numéro de cette ligne, nous obtenons la position de départ du texte, et un réaffichage complet de ce texte (corrigé de la nouvelle valeur du contraste courant) paraitra inapercu.

La fonction permettant de lire la position du curseur (en mode texte uniquement) porte le numéro 3 et est incluse dans l'interruption 10h du BIOS. Les paramètres d'entrée et de sortie sont les suivants :

Entrée :

AH = 3
BH = numéro de page

Sortie :

CH = Ligne de départ de l'aspect du curseur
CL = Ligne de fin de l'aspect du curseur

DH = Ligne du curseur (0..24)
DL = Colonne du curseur (0..79)

Le paramètre placé en BH fait référence à la possiblité des cartes graphiques de gérer plusieurs pages de texte de 4000 octets (contenant le code des caractères ASCII affichés et leur attribut de couleur). Ceci permet, par exemple, de mettre à jour un écran sans modifier l'affichage courrant et basculer sur cet écran une fois cela fait, donnant l'impression d'une modification instantanée de l'affichage. Sur le HP95, les pages n'étant pas gérées (à ma connaissance), il faudra toujours utiliser la page 0, qui correspond à l'écran actif.

Les valeurs retournées dans les registres CH et CL nous informent de l'aspect du curseur, c'est à dire son épaisseur. Si CH=7 et CL=6, le curseur a la forme de la classique barre de soulignement, si CH=7 et CL=0, il a la forme d'un pavé plein. Pour modifier l'aspect du curseur, on utilisera la fonction numéro 1 de l'interruption 10h.

Dans notre programme ce sont cependant les deux dernières valeurs qui nous intéressent. Nous n'aurons qu'à soustraire le contenu du registre DH d'une certaine valeur (8 pour notre exemple) grâce à l'instruction sub et réutiliser les mêmes registres pour repositioner le curseur grace à la fonction numéro 2, qui admet les paramètres suivants :

Entrée :

AH = 3
BH = numéro de page
DH = Ligne du curseur (0..24)
DL = Colonne du curseur (0..79)

Sortie :

Néant.

Gestion du clavier

Les fonctions gérant le clavier sont regroupées sous l'interruption 16h. Celle qui nous intéresse (la lecture d'une touche du clavier), porte le numéro 0 et admet les paramètres suivants :

Entrée :

AH = 0

Sortie :

AX = Code du caractère.

Le contenu du registre AX, en sortie, est de la forme sscc qui admet les deux cas suivants :

- Si cc et égal à zéro, ss contient le code code d'une touche spéciale (UP, F1....). ASCII de la touche correspondante.

- Si cc est différent de zéro, cc contient le code ASCII d'une touche normale ('a', ';', ENTER), et ss contient le code géographique de cette touche (le *Scan Code*).

Pour plus de détails, vous pouvez vous reporter à mon article traitant de ce sujet dans *JPC 77*.

Il est à noter que si vous ne désirez que savoir si une touche à été pressée et que vous ne désirez pas de temps d'attente, dans le cas d'un jeu vidéo par exemple, vous devrez utiliser la fonction numéro 1 de cette même interruption 16h, pour effectuer seulement un test de présence de touche en attente dans le buffer. Cette fonction admet les paramètres suivants :

Entrée :

AH = 1

Sortie :

Si le flag Z (Zéro) est à 1, aucun caractère n'est présent dans le buffer.
Si le flag Z est à zéro, un caractère est présent dans le buffer et
AX = Code du caractère.

Par exemple, vous pouvez tester l'état du flag de la façon suivante :

```
boucle :            ;début de la boucle
          .
          .
      (affichage dynamique)
          .
          .
      [redacted]  ah,1
      int   16h       ;Lecture état du clavier
      jz    boucle    ;si pas de touche en attente,
                      ; on boucle
      [redacted]  ah,0      ;sinon...
      int   16h       ;..on lit le code de touche
      cmp   ah,4800h  ;et [redacted] traite
```

Notez que, dans le cas où une touche est présente dans le buffer, vous devrez obligatoirement utiliser la fonction 0, car la fonction 1 ne retire pas la touche du buffer.

Contrôle du contraste

Seconde fonction spécifique au HP95 dans ce programme (et celle qui [redacted] intéresse le plus), la fonction 47h de l'interruption 15h controle le contraste. Il existe deux fonctions permettant soit de lire le contraste courrant, soit le changer.

Lecture du contraste :

Entrée :

AH = 47h
AL = 1

Sortie :

AL = Contraste courant

Correction du contraste :

Entrée :

AH = 47h
AL = 0
BX = Nouveau contraste

Sortie :

Aucune

Ecriture du programme

Maintenant que nous avons vu les differentes fonctions que nous utiliserons dans ce programme, nous pouvons commencer à l'écrire.

Ceux qui ont suivi mon article précédent devraient reconnaitre sans peine les premières lignes qui nous permettrons de générer un programme .COM.

Ensuite suivent les déclarations des codes de touches que nous utiliserons plus tard.

La première chaine de caractère est destinée au message d'erreur à afficher [redacted] cas d'exécution du programme sur une machine autre que le HP95.

La seconde contient le texte que nous afficherons au cours du programme.

Les variables `D_Init` et `D_curr` sont prévues pour pointer directement dans le texte. Ceci permet de mettre directement à jour (après conversion en ASCII) les valeurs des contrastes dans cette chaine. Ceci permet de faire la totalité de l'affichage (zones constantes et variables) en un seul appel système.

D'autre par vous pourrez noter que pour créer des lignes vides entre les valeurs de contraste j'ai utilisé des séquences `CR/LF/LF` (13,10,10) [redacted] non `CR/LF/CR/LF`. Ceci est tout simplement dû au fait que je n'ai pas jugé utile d'effectuer un second retour chariot (`CR`), car le premier suffit.

Après ces deux chaines de caractères, nous trouvons deux variables. La première contiendra la valeur initiale du contraste, que nous réutiliserons si [redacted] voulons abandonner notre programme par la touche ESC. La seconde contiendra la valeur courante du contraste.

Après un début d'exécution classique, nous testons si nous sommes bien sur un HP95. Dans le cas positif, nous continuons en lisant le contraste initial (en appelant le sous-programme `rd_cont`) et en le stockant sous forme numérique dans les variables prévues à cet effet et sous forme de chaine de caractères (après appel du sous-programme de conversion) dans le corps de la chaine d'affichage.

Après, l'affichage de la chaine de caractères, nous attendons la frappe d'une touche [redacted] clavier et nous préparons le code retourné pour la suite des opérations :

Les codes de touches qui ont été définis par des directives `EQU` se rapportent au contenu directement retourné par le registre AX. Dans le cas des codes ASCII standard, le Scan Code peut ne pas être constant et poser des problèmes en cas de comparaisons de ces constantes avec le contenu du registre AX. C'est pour cela que, pour ces touches, l'octet de poids fort de la constante est mis à zéro. Nous pourrons donc faire une comparaison saine [redacted] effacant le scan code (par un `xor ah,ah`) avant celle ci.

Par rapport à la liste de touches parues dans *JPC 77*, il est possible de passer de l'un à l'autre de la façon suivante :
- Si le code est inférieur à 256, il s'agit d'un code ASCII standard. Il suffira donc de le convertir [redacted] hexa décimal.
- Si le code est supérieur à 256, il s'agit d'un code étendu et nous devrons le convertir en Hexa, puis prendre les deux quartets de poids faibles et les placer comme octet de poids ford de la constante. Par exemple, la touche F4 portant le code 318, la conversion en Hexa donne 13E, ce qui donnera la constante 3E00h.

Si la touche que nous venons de frapper est RETURN ou ESC, nous sortons du programme (en replaçant le contraste initial dans le cas de ESC).

Si la touche que nous venons de frapper est UP ou DOWN, nous modifions la variable contenant le contraste courant (en prenant soin de ne pas dépasser les valeurs limites), puis nous changeons le contraste.

Ensuite, afin de conserver [redacted] affichage stable nous récupérons la position du curseur et nous le faisons remonter de 8 lignes (taille de notre texte).

Juste avant de retourner [redacted] l'affichage, nous convertissons la nouvelle valeur du contraste en ASCII, afin d'avoir la bonne valeur dans AX, qui sera utilisé dans la première ligne de la boucle.

Les deux sous-programmes suivants, permettent d'accéder aux fonctions de gestion du contraste. Dans les deux cas (en entrée ou en sortie), la valeur du contraste est placée en AL, ce qui est plus logique par rapport à la fonction 47h, qui utilise deux registres différents en entrée et en sortie pour la valeur du contraste. Notez que si vous voulez porter ce programme sur [redacted] autre ordinateur (je pense à toi Guy !), il ne faudra modifier que ces deux fonctions.

La dernière fonction permet de convertir le contenu du registre AL en deux caractères ASCII dans le registre AX. Ceci est fait en utilisant l'instruction AAM qui peut effectuer une conversion BCD sur le contenu de AL vers AX. Par exemple, si nous avons 0Ch (12 en décimal) dans AL, nous obtiendrons directement 0102h dans AX. Nous pourrons additionner 3030h à ce registre, ce qui nous permettra d'y obtenir les caractères '12'. Afin d'avoir un affichage plus agréable, nous replaçons, si nécessaire, le premier caractère '0' par [redacted] espace. Enfin, nous échangeons les deux octets, car ce registre devant être écrit directement en mémoire, ceux ci y seront écrit dans l'ordre inverse de celui du registre (comme pour le HP71).

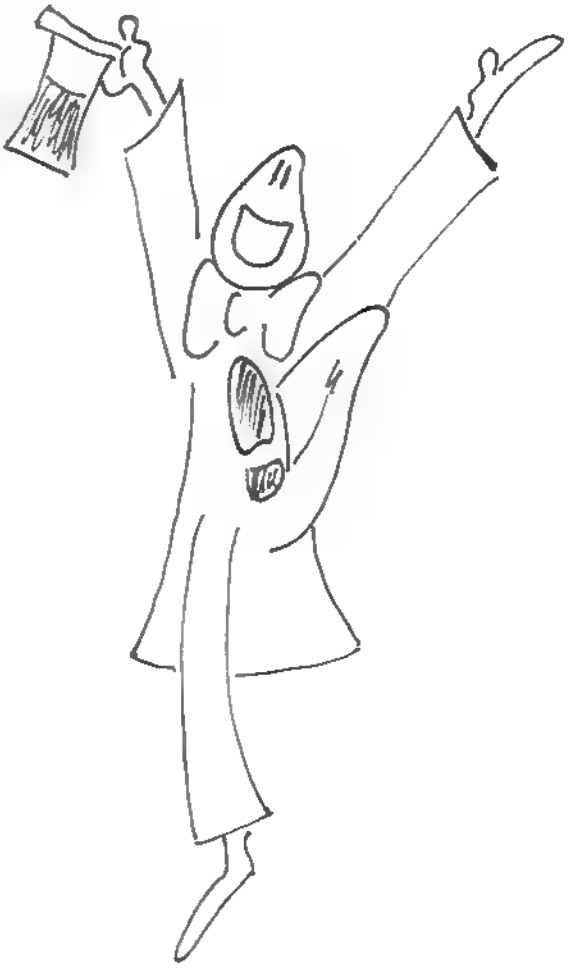
Ces explications étant terminées, vous pouvez maintenant observer le listing complet. Vous pourrez trouver l'image du fichier compilé dans le *Coin des Codes*.

```
        DOSSEG
code    segment para 'CODE'
        assume cs:code
        ORG     100h

start:  jmp     strt            ;première inst. exécutée

K_ESC   EQU     001Bh           ;definition des codes de
K_ENTER EQU     000Dh           ;  touches
K_UP    EQU     4800h
K_DOWN  EQU     5000h

Not_95  [redacted]      "This program runs only on [redacted] HP95LX"
        DB      13,10,"$"

Dispstr DB      "HP95LX Contrast control",13,10,10
        DB      "Initial contrast = "
D_init  [redacted]      "XX",13,10
        DB      "Current contrast = "
D_curr  DB      "XX",13,10,10
        [redacted]      "Contrast :  UP = +1   DOWN = -1 ",13,10
        [redacted]      "ESC = Exit without Change",13,10
        DB      "ENTER = Exit with Change",13,10,"$"

C_init  DB      (?)             ;Contraste initial
C_curr  [redacted]      (?)             ;Contraste courant

strt:   mov     bx,cs           ;nécessaire pour la
        mov     ds,bx           ; fonction 09h

        mov     ax,4dd4h        ;vérification d'exécution
        int     15h             ; sur HP95
        cmp     bx,'HP'
        jne     nok_95
        cmp     ch,1
        je      ok_95           ;si hp95, [redacted] passe
nok_95: [redacted]      dx,OFFSET Not_95
        mov     ah,09h
        int     21h
        jmp     exit            ;sortie du programme

ok_95:  call    rd_cont         ;acquisition contraste
        mov     [C_init],al     ;stockage dans variables
        mov     [C_curr],al
        call    htoa            ;conversion en texte
        mov     WORD PTR [D_init],ax ;sto dans texte
boucle: mov     WORD PTR [D_curr],ax
        mov     dx,OFFSET Dispstr
        mov     ah,09h
        int     21h             ;affichage du texte

wait_key :
        mov     ah,0
        int     16h             ;Lecture du clavier
        cmp     al,0            ;code étendu ?
        je      ok
        xor     ah,ah           ;on efface le scan code
ok:     cmp     ax,K_ENTER      ;touche ENTER ?
        je      exit            ; oui, sortie
        cmp     ax,K_ESC        ;touche ESC ?
        je      c_esc           ; oui, traitement ESC
        cmp     ax,K_UP         ;touche UP ?
        je      c_up            ; oui, traitement UP
        cmp     ax,K_DOWN       ;touche DOWN ?
        je      c_down          ; oui, traitement DOWN
        jmp     wait_key        ;autres cas, on boucle

c_up:   cmp     BYTE PTR [C_curr],15
        je      ovrflw          ;si contraste = 15,
                                ; pas d'incrémentation
```

```
        inc     BYTE PTR [C_curr] ;incrémente contrast
ovrflw: jmp     redisp

c_down: cmp     BYTE PTR [C_curr],0
        je      redisp        ;si contraste = 0,
                              ; pas de décrémentation
        dec     BYTE PTR [C_curr] ;décrémente contrast
redisp: mov     al,[C_curr] ;lecture contraste
        call    st_cont       ;correction contraste
        mov     ah,03h        ;lecture position du
                              ; curseur
        mov     bh,0
        int     10h
        sub     dh,8          ;calcul ligne début texte
        mov     ah,2          ;repositionnement curseur
        mov     bh,0
        int     10h
        mov     al,[C_curr] ;lecture contraste
        call    htoa          ;conversion [redacted] texte
        jmp     boucle        ;retourne vers affichage

c_esc:  [redacted]  al,[C_init] ;lecture contraste init.
        call    st_cont       ;on remet tout [redacted] ordre
exit:   mov     ah,4ch        ;retour au DOS
        int     21h

rd_cont PROC NEAR             ;lecture du contraste
        mov     ax,4701h
        int     15h
        ret
rd_cont ENDP

st_cont PROC NEAR             ;correction du contraste
        mov     bl,al
        xor     bh,bh
        [redacted]  ax,4700h
        int     15h
        ret
st_cont ENDP

htoa    PROC NEAR             ;conversion en ASCII
        [redacted]
        add     ax,3030h
        cmp     ah,'0'
        jne     htoa_1
        mov     ah,' '
htoa_1: xchg    ah,al
        ret
htoa    ENDP

Code    ends
        END     start         ;fin du programme
```

Pour terminer, j'ajouterai que le programme compilé en .COM faisant moins de 512 octets (taille d'un secteur de disquette), il n'y [redacted] théoriquement que peu d'intérêt à l'optimiser. Cependant, j'avoue ne pas m'être beaucoup fatigué pour l'écrire et j'espère que vous trouverez là un bon exercice, en cherchant à faire une meilleure version, pour commencer à exploiter les quelques connaissances que j'ai déjà tenté de vous donner !

Jacques Belin (123)



LE COIN DES CODES

La compilation de certains programmes, tels ceux écrits en assembleur, nécessitent souvent un logiciel que ▬ possèdent pas tous nos lecteurs. Le *Coin des Codes* permet de résoudre ce problème.

Note importante :

Même si la présentation des listings est identique, le traitement de ceux-ci est différente suivant le programme d'entrée et la machine de destination. Chaque listing est prévu pour être entré sur ▬ machine de destination. Par exemple, ne tentez pas d'entrer un programme HP48 dans un fichier MS-DOS à l'aide du programme MAKEDOS. Vous obtiendrez un fichier que vous ne pourrez pas transférer dans la HP48.

Programmes HP48

Par rapport aux méthodes habituelles sur HP48, notre méthode effectuant un calcul local du checksum en fin de chaque ligne permet de faciliter la recherche d'erreurs, par rapport à une même recherche dans une chaîne de plusieurs centaines d'octets.

Ceux qui n'ont pas encore de programme assembleur pourront procéder de la manière suivante:

- par mesure de sécurité sauvegardez vos programmes et fichiers, éventuellement verrouillez vos cartes RAM pour devenir ROMs.
- tapez le programme ASSCOD.

ASSCOD

7B74h
879.5 octets

```
« RCLF HEX 64 STWS -2 SF 1 CF "" 'tmpcod' STO
  "nombre d'octets"  "" INPUT OBJ→ 2 * 16
  DUP2 / IP 3 ROLLD ▬ DUP2
  IF
  THEN 1 +
  ▬ 3 ROLLD 1 + 4 ROLL
  # 0h → n r f s
    « 1 SWAP
      FOR i
        DO
          "ligne        "
          "00" i R→B →STR ▮ OVER SIZE 1 -
          SUB + DUP SIZE DUP 2 - SWAP ▬ + DUP
          "                    ▮ chaîne commençant
          chaine" +            ▮ par ← (newline)
          "                    ▮ chaîne commençant
          ___ ___ ___ ___"     @ par ← (newline)
                               ▮ et composée de 4
                               ▮ groupes de 4
                               @ CHR 95 obtenus par
                               ▮ α shift bleu ▮
                               @ 1 espace séparant
                               ▮ 2 groupes
          n i <
          IF
          THEN ▮ r ▬
          END + ▮ FC?C
          IF
          THEN { "" α }
          ELSE ROT
          END INPUT 0 OVER SIZE 1 SWAP
          FOR j OVER j DUP SUB NUM ▮ * +
          NEXT s + DUP ▮ FFFh AND "#"
          "                    @ "somme de controle"
          somme de controle    ▮ précédée et suivie
          ___"                 ▮ de ← (newline)
                               ▮ puis 3 CHR ▬
                               ▮ (α shift bleu x)
          6 ROLL SWAP + { "" ▬ }
          INPUT + OBJ→ ==
          IF
          THEN 1
          ELSE DROP 1000 .5 BEEP 1 SF 0
          END
        UNTIL
        END 's' STO 'tmpcod' DUP RCL
        ROT + SWAP STO
      NEXT f STOF
    »
  tmpcod
  WHILE DUP " " POS DUP
  REPEAT DUP2 1 SWAP 1 - ▬ ▮ ROLLD 1 + MAXR
     SUB +
  END DROP
  "GROB 8 "                    @ si vous avez ASC→
    OVER SIZE 2 / " " +        @ JPC 79 page 14
    + SWAP + STR→              ▮ vous pouvez remplacer
    ▮ 4017h SYSEVAL            @ ces lignes
    ▮ 56B6h SYSEVAL            @ par ASC→
    DROP NEWOB                 @ (plus rapide)
»
```

Donc à partir de maintenant pour tout assemblage de chaîne de codes procédez de manière suivante :

1- lancez le programme ASSCOD.
2- donnez le nombre d'octets (1/2 oct. compris).
3- tapez chaque ligne de codes avec ▬ espace entre chaque groupe de 4 caractères.
4- tapez la somme de contrôle. S'il y a erreur la ligne sera réaffichée pour correction après émission d'un BEEP.

5- stockez le programme assemblé dans la variable donnée en tête.
6- si tout s'est bien déroulé vous pouvez purger tmpcod qui contient la chaîne de codes.

Programmes MS-DOS

Afin d'être utilisé par tous, ce programme est destiné à être écrit en GWBASIC. Il devrait cependant être facile de le convertir pour un autre programme (QBASIC, Turbo BASIC...).

Mode d'emploi :

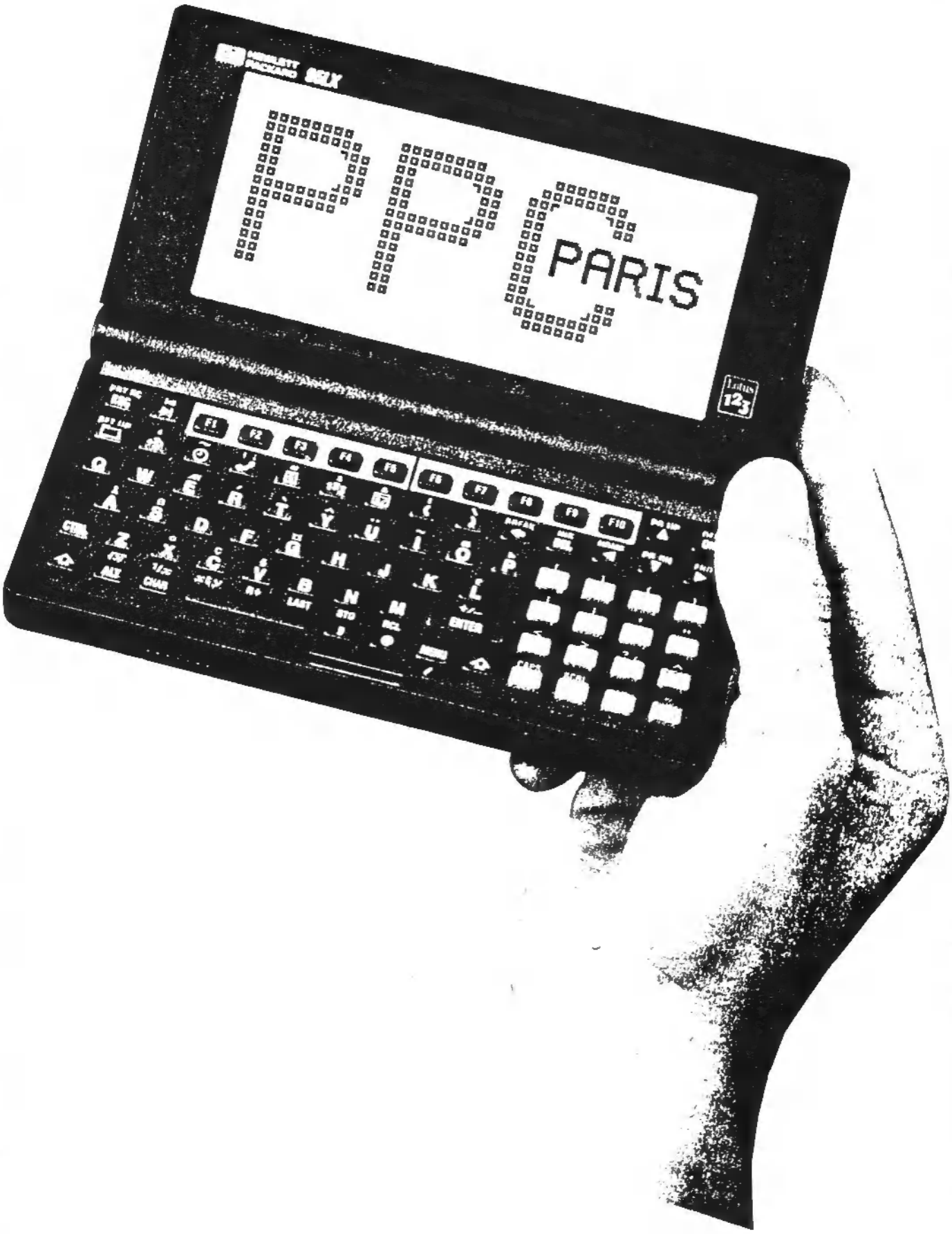
1- Lancer le programme : GWBASIC MAKEDOS.BAS
2- Entrer le nom du fichier destination.
3- Entrer la taille du fichier.
4- Entrer les listes de codes puis le checksum (en prenant soin d'entrer les codes héxadécimaux en majuscules). En cas d'erreur corriger la ligne, en prenant soin de placer le curseur après le dernier caractère avant de taper sur la touche d'entrée.
5- Une fois que toutes les lignes sont entrées, sortir du GWBASIC en exécutant la commande SYSTEM. Le nouveau programme est immédiatement disponible.

Note : La taille du fichier résultant peut être supérieure d'un octet à ce qui est affiché dans le listing. Cela n'est pas un problème.

Programme MAKEDOS.BAS

```
 10 INPUT "Nom du fichier : ",NOM$ : INPUT "Nombre d'octets : ",N : N=N*2
 20 OPEN NOM$ FOR OUTPUT AS #1 : CLOSE #1 : KILL [redacted]
 30 OPEN "bin.tmp" FOR OUTPUT AS #1 : S=0 : P$="---- ---- ---- ----"
 40 NLINES=N\16 : LENLAST=(N [redacted] 16)+((N [redacted] 16)\5)
 50 IF (N MOD 16)=0 THEN NLINES=NLINES-1 : LENLAST=LEN(P$)
 [redacted] FOR X=0 TO NLINES
 70   IF X=NLINES THEN P$=LEFT$(P$, LENLAST)
 80   C$=P$
 90   X2$="00"+HEX$(X) : PRINT RIGHT$(X2$,3);":";
100   Y=CSRLIN : LOCATE Y,6 : PRINT C$; : LOCATE Y,6 : INPUT "",C$ : Y=Y-1
110   LOCATE Y,27 : PRINT " sm = ---" : LOCATE Y,33 : INPUT "",D$
120   M=S
130   FOR Z=1 TO LEN(C$)
140     IF MID$(C$,Z,1)<>" " THEN M=(M+((Z-(Z\5))*ASC(MID$(C$,Z,1)))) MOD 4096
150   NEXT Z
160   D2$="00"+HEX$(M) : D2$=RIGHT$(D2$,3)
170   IF D2$<>D$ THEN PRINT "Erreur de somme" : BEEP : GOTO 90
180   FOR Z=1 TO LEN(C$) STEP 2
190     IF MID$(C$,Z,1)=" " THEN Z=Z-1 : GOTO 230
200     CH=ASC(MID$(C$,Z,1))-48 : IF CH>9 THEN CH=CH-7
210     CL=ASC(MID$(C$,Z+1,1))-48 : IF CL>9 THEN CL=CL-7
220     PRINT#1,CHR$((16*CH)+CL);
230   NEXT Z
240   S=M
250 NEXT X
260 CLOSE #1 : NAME "bin.tmp" AS [redacted] : END
```

```
FFACTS                    (HP48)
# B80Fh              264 octets

     0123 4567 89AB CDEF   ▬

000: D9D2 0ECE 819F F30D   FDA
001: 9D20 8813 07D8 A283   D2B
002: 7A22 C230 D4FA 2FC7   CD7
003: A257 B302 C230 3392   8E8
004: 0400 0000 0000 0001   283
005: 00A8 A20B 536A 2170   F31
006: 2AC8 1881 3058 CC11   C75
007: 78A2 3991 6D9D 20C4   A9B
008: ▬▬ B261 B213 0756     702
009: 609B C268 A056 EF9A   6B4
00A: 218B A2CB 9A2D CC26   68E
00B: 479A 2344 A2CB 9A29   53E
00C: 0BA2 18BA 2479 A237   2E1
00D: FA2A EC81 F316 12C2   09B
00E: 30F6 E30C 1C16 33F0   E40
00F: 6AEC 81CC D20B 0100   B6C
010: 8FD5 F30D E06A F2D6   B21
011: 10A8 FB97 6014 3131   729
012: 1791 3310 1131 D210   26E
013: 8E62 B155 1119 1341   DF2
014: 18D7 D206 05AF 0152   AE9
015: 111A AF18 2281 E832   82B
016: 50A7 8A74 8AEC E07A   825
017: 7915 4116 BA92 8128   501
018: 1281 2812 04CF 59B8   341
019: AAD0 1441 10E4 1001   EAE
01A: 1ACE 10A8 AEF8 2007   CE1
01B: CED5 81ED 7119 C913   AC3
01C: 4061 5A01 5F01 5901   6C8
01D: 5C01 7018 0CF5 7E07   4BE
01E: 135C 9134 3103 15F0   0EE
01F: 15C1 1811 C0CD 5FE8   096
020: D341 50B2 130B 2130   C3B

FFACTSX                   (HP48)
▬ 7561h            265.5 octets

     0123 4567 89AB CDEF   sm

000: D9D2 0ECE 819F F30D   FDA
001: 9D20 8813 07D8 A283   D2B
002: 7A22 C230 D4FA 2FC7   CD7
003: A257 B302 C230 3392   8E8
004: 0500 0000 0000 0001   285
005: 00A8 A20B 536A 2170   F33
006: 2AC8 1881 3058 CC11   C77
007: 78A2 3991 6D9D 20C4   A9D
008: EA28 B261 B213 0756   704
009: 609B C268 A056 EF9A   6B6
00A: 218B A2CB 9A2D CC26   690
00B: 479A 2344 A2CB 9A29   540
00C: 0BA2 18BA 2479 A237   2E3
00D: FA2A EC81 F316 12C2   09D
00E: 30F6 E30C 1C16 33F0   E42
00F: 6AEC 81CC D20E 0100   B92
010: 8FD5 F30D E06A F2D6   B47
011: 10A8 FB97 6014 3131   74F
012: 1791 3310 1131 D210   294
013: 8E62 A155 1119 1341   E13
014: 18D7 D206 05AF 0152   B0A
015: 111A AF18 2281 E832   84C
016: 50A7 8A74 8AEC E07A   846
017: 7915 4116 AA92 8128   519
018: 1281 2812 8120 4CF5   24B
019: 688A AD01 4411 0E41   EFB
01A: 0011 ACE1 0A8A EC82   E65
01B: 007C ED58 1ED7 119C   D67
01C: 9134 0615 A015 F015   9BB
01D: 9015 C017 0180 CF57   741
01E: E071 35C9 1343 1031   2AA
01F: 5F01 5C11 811C 0CD5   0C0
020: FE8D 3415 0B21 30B2   D69
021: 130                   E9D

REV                       (HP48)
▬ 4FB0h             60.5 octets

     0123 4567 89AB CDEF   sm

000: D9D2 0ECE 81D0 040D   E3B
001: 9D20 BB72 6CCD 2015   C27
002: 0008 F146 608F B976   A4E
003: 0D81 4313 1179 1371   5D3
004: 35C9 C913 4819 F154   313
005: 1181 14F1 4A14 914C   069
006: 171C D5BE 8D34 150B   EC3
007: 2130 B213 0           7C5

TIMED                     (HP48)
▬ 1C05h               78 octets

     0123 4567 89AB CDEF   ▬

000: D9D2 0530 26CB 9161   D01
001: 8B02 AAF0 2930 45AD   B50
002: 846A AC35 47A2 003D   8EE
003: 4303 D43B 2130 0D47   5A6
004: 018B E00F 516E 8F60   40C
005: 18BE 07E3 1690 DA1D   2E5
006: 5345 EEFE 0EF9 A233   23F
007: 9207 9900 0000 0007   CBB
008: 5301 89A2 6831 6AAC   BB1
009: 3579 470B 2130        BA2

CONTRAST.COM              (MS-DOS)
                       397 octets

     0123 4567 ▬▬ CDEF   sm

000: E9C9 0054 6869 7320   C1E
001: 7072 6F67 7261 6D20   ▬
002: 7275 6E73 206F 6E6C   7DE
003: 7920 6F6E 2061 6E20   4E3
004: 4850 3935 4C58 0D0A   2A9
005: 2448 5039 354C 5820   F01
006: 436F 6E74 7261 7374   BB1
007: 2063 6F6E 7472 6F6C   A9D
008: 0D0A 0A49 6E69 7469   886
009: 616C 2063 6F6E 7472   624
00A: 6173 7420 3D20 ▬▬     ▬
00B: 0D0A 4375 7272 656E   FEC
00C: 7420 636F 6E74 7261   CF1
00D: 7374 203D 2058 580D   9FC
00E: 0A0A 436F 6E74 7261   748
00F: 7374 203A 2020 5550   29B
010: 203D 202B 3120 2020   D91
011: 444F 574E 203D 202D   B23
012: 3120 0D0A 4553 4320   ▬
013: 3D20 4578 6974 2077   32F
014: 6974 686F 7574 2043   F88
015: 6861 6E67 650D 0A45   D04
016: 4E54 4552 203D 2045   8FF
017: 7869 7420 7769 7468   5B5
018: 2043 6861 6E67 650D   341
019: 0A24 0000 ▬▬ 8EDB     382
01A: B8D4 4DCD 1581 FB50   225
01B: 4875 0580 FD01 740A   F75
01C: BA03 01B4 09CD 21E9   DE7
01D: 8000 E881 002E A2CA   C80
01E: 012E A2CB 01E8 8600   9DF
01F: 2EA3 5601 2EA3 6D01   748
020: BA29 01B4 09CD 21B4   55B
021: 00CD 163C 0074 0232   120
022: E43D 0D00 7454 3D1B   ECA
023: 0074 483D 0048 7407   AF7
024: 3D00 5074 12EB E02E   967
025: 803E CB01 0F74 052E   72C
026: FE06 CB01 EB0E 902E   627
027: 803E CB01 0074 052E   310
028: FE0E CB01 2EA0 CB01   199
029: E829 0084 0387 00CD   F9B
02A: 1080 EE08 B402 B700   C89
02B: CD10 2EA0 CB01 E81D   ▬
02C: 00EB 992E A0CA 01E8   AA0
02D: 0A00 B44C CD21 B801   867
02E: 47CD 15C3 8AD8 32FF   85D
02F: B800 47CD 15C3 D40A   6DE
030: 0530 3080 FC3D 7502   31E
031: B420 86E0 C3          F36
```


HEWLETT PACKARD HP-41CX
PPC
PARIS
Lotus 123

PPC PARIS
B.P. 604
75028 Paris Cedex 01
France

Paris, le 16 Janvier 1993

Cher Ami,

Nous avons bien reçu votre fiche d'inscription à notre réunion des 30 et 31 Janvier 1993 et vous remercions de votre venue.

Vous trouverez joints à cette lettre divers documents vous permettant de trouver le lieu de réunion.

En cas de problème, deux numéros de téléphone vous seront accessibles. Le premier (42 03 00 47) correspond au centre Jean Verdier, et sera accessible pendant les heures prévues pour la réunion (9h-19h les samedi et dimanche). En dehors de ces heures, vous pourrez tenter de me joindre à mon numéro personnel (42 37 72 67). Si je suis absent, vous pourrez laisser à message sur le répondeur, que j'interrogerais régulièrement.

N'hésitez pas à nous contacter si vous avez besoin d'aide, par exemple pour transporter des objets lourds ou encombrants.

Amicalement,

Le président
Jacques Belin,

P.S: Pouvez vous amener, si vous l'avez encore, votre carte HP-IL -> Synthese vocale?

REUNION DU 10eme ANNIVERSAIRE DE PPC PARIS

30-31 Janvier 1993

Centre Culturel Jean Verdier
11 rue de Lancry, Paris 10

Programme non définitif
(Autres présentations non encore confirmées)

Samedi 30 Janvier :

9h-10h :	Enregistrement des participants.
10h-13h :	Rondes 1 et 2 du tournoi de programmes. (en parallèle : présentations et réunions informelles, et stands éditeurs, distributeurs et clubs).
13h-14h :	Pause déjeuner.
14h-16h :	Assemblée générale PPC Paris.
16h-16h30 :	Pause café
16h30-17h30 :	Présentations en séance plénières : - Une carte vous permet de stocker jusquà 20 Mo sur votre HP95, par Leon Malmed (SunDisk Corp). - Nouveaux produits pour HP48 et HP95. Par Jacques Belin
17h30-19h :	Présentations à préciser
19h30-???? :	Banquet anniversaire.

Dimanche 31 Janvier :

9h-13h :	Rondes 3, 4 et 5 du tournoi de programmes (en parallèle : présentations et réunions informelles, stands distributeurs et clubs)
13h-14h :	Pause déjeuner
14h-15h30h :	Sessions parallèles de développement sur HP48 et HP95. Intervenants : Jacques Belin (HP95), Jean Francois Garnier ou Guy Toublanc (HP48)
15h30-16h :	Pause Café
16h-17h :	Présentations en séance plénière : - Commande de train electrique avec une HP48, par Robert Pulluard. - Projection de la cassette vidéo "Corvallis Plant Tour"
17h-18h :	Table Ronde avec les personalités présentes.
18h-19h :	Remise des prix du tournoi et Pot d'adieu

Note Importante : L'accès à l'assemblée Générale est libre et gratuit.

Le Journal JPC est le bulletin de liaison entre les membres de l'Association "PPC Paris", régie par la loi de 1901 et indépendante de tout constructeur ou société commerciale. Le Club est éditeur de JPC, et son siège social est au 56, rue Jean-Jacques Rousseau, 75001 Paris.

La maquette de ce numéro a été préparée et réalisée par Jacques Belin et Asdin Aoufi.

Les dessins sont de Jean-Jacques Dhénin et Paul Courbis.

Les informations et programmes parus dans ce journal sont publiées "Tels quels" et ne peuvent en aucun cas engager la responsabilité de Hewlett-Packard ou de PPC Paris. Hewlett-Packard se réserve le droit de ne pas répondre aux questions concernant le sujet de certains articles.

Les programmes publiés peuvent être utilisés librement. Cependant, ils ne peuvent être vendus ou fournis dans un ensemble commercialisé, sous quelque forme que ce soit, sans l'accord écrit de l'auteur ou de PPC Paris.

Directeur de la publication : Jacques Belin
Numéro ISSN : 0762 - 381X

Veuillez adresser toute correspondance à :
PPC Paris, BP 604, 75028 Paris Cedex 01.